Hausbacher

TODO SOBRE EL PROCESADOR



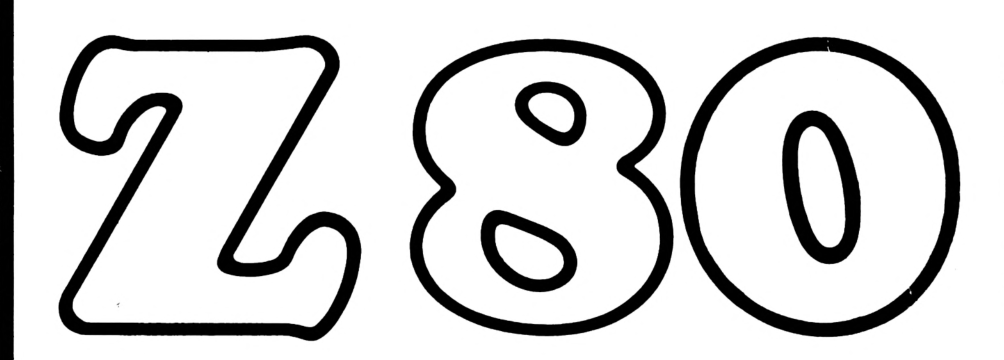
Técnica y Programación

UN LIBRO DATA BECKER

EDITADO POR FERRE MORET, S.A.

Hausbacher

TODO SOBRE EL PROCESADOR



Técnica y Programación

UN LIBRO DATA BECKER

EDITADO POR FERRE MORET, S.A.

La dirección y colaboradores de Ferré Moret deseamos atestiguar nuestro recuerdo por Antonio LEMKHUL PEREZ, compañero trágicamente desaparecido en accidente, dedicándole este libro, que es el último en que cooperó.

Imprime: APSSA, ROCA UMBERT, 26 - L'HOSPITALET DE LL. (Barcelona)

Depósito legal: B-10511/86

ISBN 84 - 86437 - 32 - 6

Este libro ha sido traducido por Montserrat RUBBEL PONS experta conocedora del Commodore 64.

Advertencia importante

Los circuitos programas y procedimientos reproducidos en este libro, son divulgados sin tener en cuenta el estado de las patentes. Están destinados exclusivamente al uso amateur o docente, y no pueden ser utilizados para fines comerciales.

Todos los circuitos, datos técnicos y programas de este libro, han sido elaborados o recopilados con el mayor cuidado por el autor y reproducidos utilizando medidas de control eficaces. No obstante, es posible que exista algún error. Ferré Moret, S.A. se vé, por tanto, obligada a advertirles, que no puede asumir ninguna garantía, ni responsabilidad jurídica, ni cualquier otra responsabilidad sobre las consecuencias atribuibles a datos erróneos. El autor les agradecerá en todo momento la comunicación de posibles fallos.

I N D I C E

Las Fig. 1.1 hasta 1.3, 1.6, 1.7, 2.2 hasta 2.16, 5.4, 5.7, 6.15 hasta 6.20 y 7.1 hasta 7.6 han sido impresas con el gentil consentimiento de Zilog Inc., USA. La Fig. 5.3 ha sido copiada con el gentil consentimiento de Intel Corporation, USA.

1. Hardware del Z80-CPU

1.1 Generalidades

A pesar de que el microprocesador de 8 bits del Z80 se encuentra ya en el mercado desde 1977 y que desde entonces con tecnologías mejoradas de producción se han podido hacer realidad microprocesadores de 16 y de 32 bits, dicho Procesador puede actualmente hacerse todavía fuerte en el mercado y seguirá siéndolo en un futuro. Para ello existen dos motivos fundamentales: En primer lugar porque se trata de un producto madurado y asequible, con una elevada capacidad de rendimiento, y en segundo lugar porque pueden cubrirse muchos campos de aplicación de modo satisfactorio, para los cuales las máquinas de 16 y de 32 bits de gran calibre serían demasiado costosas. El Z80 puede encontrarse tanto en ordenadores personales (Osborne, Sirius MTX 512, etc.) y en Homecomputer (por ejemplo de la serie MSX), como en controles industriales (por ejemplo Feltron) y mandos de aparatos (por ejemplo aparatos de navegación para satélites). Según sea la cantidad de datos exigida, se encuentran a disposición el Z80-CPU (2,5 MHz de frecuencia de reloj) o bien el Z80A-CPU (4 MHz de frecuencia de reloj). Para algún que otro usuario podría ser de interés la compatibilidad ascendente hacia el software con el procesador 8080 (Intel). El microprocesador Z80 ha sido creado ségún la tecnología NMOS y requiere una tensión de alimentación de 5 V ± 5 %. El consumo de corriente es de 150 mA (Z80) y de 200 mA (Z80A), como máximo. Entretanto la tecnología CMOS se ha ido manteniendo en amplios campos, de modo que también existen dos versiones del Z80 en CMOS: Desde 1980 el Z80 es compatible con el software NSC 800 y desde hace poco el HD 64180 de Hitachi. Este último modelo contiene en el chip, además de la CPU, las siguientes unidades:

MMU : Unidad de tratamiento de memoria para máx. 512 K-bytes.

DMAC : Controlador DMA de dos canales.

TIMER : Dos indicadores de tiempo de 16 bits.

SIO : Tres interface serie.

También se pueden implementar instrucciones adicionales. El HD 64180 consume por ejemplo con una frecuencía de reloj de 6 MHz una corriente de sólo 15 mA. Además del escaso consumo de energía eléctrica se han formulado de forma más amplia las tolerancias de la alimentación de tensión, la estabilidad es mayor, es mayor, etc. Lo que aquí queremos es tratar en primer lugar el Z80-CPU. Junto con los componentes periféricos del Z80 pueden construirse muy fácilmente sistemas de gran rendimiento. Es evidente que puedan utilizarse también componentes periféricos de otros constructores; sin embargo éstos no están equipados con las cómodas posibilidades del vector de interrupciones de los componentes del Z80.

Con el fin de no sobrecargar al lector con un lastre innecesario de sabiduría, la representación siguiente detallará únicamente hasta el punto de poder llegar a una comprensión práctica. El resto será tratado como caja negra (Black Box) y únicamente descrito de forma funcional.

1.2 Arquitectura del sistema

Cada sistema de microordenador (Z80) consta de tres partes elementales:

1.) CPU (Central Processing Unit) - Unidad Central de Proceso.
2.) Memoria.
3.) Componentes periféricos.

Un ejemplo para un sistema mínimo del Z80 se muestra en la Fig. 1.1.

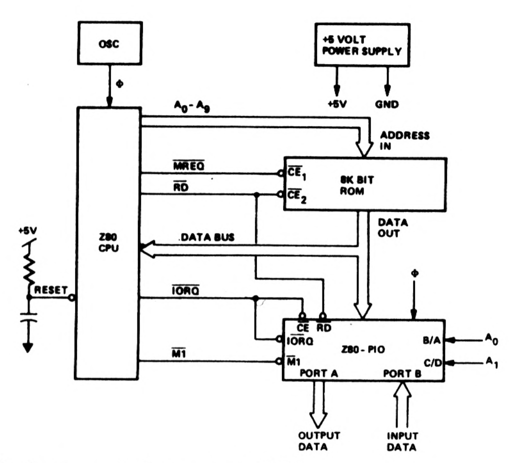


Fig. 1.1 Sistema mínimo del Z80

Un oscilador abastece a la CPU con la cadencia del sistema, la cual se requiere para la coordinación de los diversos procesos dentro de la misma.

La CPU recoge las instrucciones de la memoria (ROM) y las ejecuta.

La memoria (ROM) contiene instrucciones para la CPU (programa). La mayoría de los sistemas utilizan también RAM's como memoria; éstas contienen generalmente datos

(sistemas de control) o bien son cargadas externamente mediante programas (por ejemplo sistemas de desarrollo).

El Z80-PIO sirve para la comunicación en paralelo con aparatos periféricos y es un ejemplo de componente periférico.

La mayoría de los sistemas del Z80 contienen más componentes que el sistema mínimo de la Fig. 1.1. Frecuentemente deben colocarse buffers en los buses, reforzarse las señales mediante pasos excitadores y seleccionarse los componentes con decodificadores.

Las distintas unidades de un sistema se encuentran unidas mediante líneas comunes. Según puntos de vista funcionales dichas líneas pueden recopilarse en tres grupos:

1.) Bus de direcciones: Este es alimentado por la CPU y define la fuente y el destino de los datos a ser transferidos (16 líneas).

2.) Bus de datos: Sirve para el transporte de datos entre la CPU y los distintos componentes (8 líneas).

3.) Bus de control: Sirve para la sincronización del sistema (13 líneas).

1.3 Descripción de los Pins del Z80-CPU

El Z80-CPU se encuentra ubicado en una caja de 40 pins Dual-In-Line. La ocupación de los pins se muestra en la Fig. 1.2.

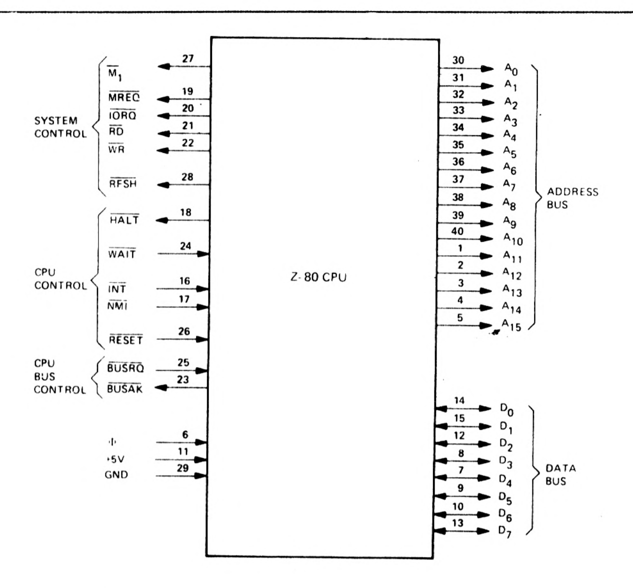


Fig. 1.2. Ocupación de los pins del Z80-CPU

A0-A15 Estas 16 líneas forman en conjunto el bus de direcciones. Con el mismo pueden formarse 2^{16} = 65.536 direcciones. Así pues podrá direccionarse un área de memoria de 64 K-bytes. Los componentes periféricos (componentes de entrada y de salida) podrán únicamente ser direccionados mediante las 8 líneas inferiores (A0-A7) (máx. 2^{8} = 256 aparatos). Finalmente las siete líneas inferiores suministrarán, durante el tiempo de "refresh", las direcciones de refresco para las RAM's dinámicas eventualmente conectadas. El A0 suministrará el bit de menor valor (ésto equivale para todas las conexiones del componente de sistemas del Z80 y registros).

D0-D7	Estas 8 líneas sirven para el transporte de datos entre la CPU y los distintos componentes del sistema. Estas forman el bus de datos.
M1	(Machine Cycle 1) Esta salida indica que la CPU está leyendo un código de operación de la memoria. Además señala a la CPU con M1 e IORQ (ambas mientras M1 se encuentra en estado lógico 0) la confirmación de un interrupt.
MREQ	(Memory Request) Esta salida es lógico 0, cuando en el bus de direcciones se encuentra una dirección de memoria válida.
IORQ	(Input / Output Request) Esta salida es lógico 0, cuando en la mitad inferior del bus de direcciones (A0-A7) se encuentra una dirección de entrada / salida válida. IORQ forma junto con M1 una señal de confirmación del interrupt.
RD	(Read) Esta salida es lógico 0, cuando la CPU desea leer datos de la memoria o bien de un componente periférico.
WR	(Write) Esta salida es lógico 0, cuando sobre el bus de datos hay una fecha válida que deberá ser escrita por la CPU en una memoria o en un componente periférico.
RFSH	(Refresh) Esta salida es lógico 0, cuando la CPU ha emitido una dirección de refresco válida para RAM's dinámicas de las líneas A0-A6.

HALT (Halt state) Esta salida es lógico 0, cuando la CPU ha realizado una instrucción de PARO hasta que tenga lugar un INT o un NMI. Durante este tiempo la CPU realizará instrucciones NOP que no ocasionan otra cosa que seguir emitiendo direcciones de refresco.

WAIT (Wait) Si esta entrada es lógico 0, la CPU sabrá que el componente de memoria direccionado o bien el componente periférico no estará preparado para una transmisión de datos e irá intercalando ciclos de espera hasta que esta entrada sea lógico 1. Así pues, dicha entrada sirve para la sincronización con componentes más lentos.

INT (Interrupt Request) Si esta entrada es lógico 0 (o sea que un componente periférico ha requerido una interrupción), la entrada BUSRQ es lógico 1 y el interrupt flip-flop IFF1 interno, controlado por el software, es lógico 1, la CPU aceptará al final de la instrucción en curso el interrupt (M1 e IORQ mientras que M1 es lógico 0). Las distintas modalidades de interrupt serán comentadas más detenidamente en el capítulo 4.

NMI (Non Maskable Interrupt) Esta entrada del interrupt tiene prioridad ante la entrada del INT y no es "mascarable", es decir que este interrupt también se aceptará, si el IFF es lógico 0, o sea interrupts que se encuentren bloqueados por el software. Si esta entrada se convierte en lógico 0 (flanco negativo), la CPU aceptará el interrupt al final de la instrucción en curso (M1 e IORQ mientras que M1 es lógico 0)

y se ramificará a la dirección 0066H.

RESET Si esta entrada es lógico 0, la CPU será llevada a un estado inicial definido. Visto detalladamente ocurre lo siguiente:

1.) El contenido del registro de interrupt será 00H.
2.) El contenido del registro de refresh será 00H.
3.) La CPU se pasará a la modalidad de interrupt 0.
4.) IFF1 e IFF2 serán postergados (lógico 0).
5.) El bus de dirección y de datos serán de alto ohmiaje durante el reset.
6.) Todas las líneas de control de salida pasarán a un estado inactivo.

BUSRQ (Bus Request) Si esta entrada es lógico 0, la CPU aceptará al final del ciclo de máquina en curso la exigencia DMA y desplazará el bus de datos, el bus de dirección y las líneas de control MREQ, IORQ, RD y WR a un estado de alto ohmiaje.

(Cadencia de sistema) Cadencia de una fase con nivel TTL. (330 ohmios - pull up resistencia contra + 5 V).

1.4 Registro del Z80-CPU

Los registros internos y accesibles para el programador del microprocesador Z80 están representados en la Fig. 1.3.

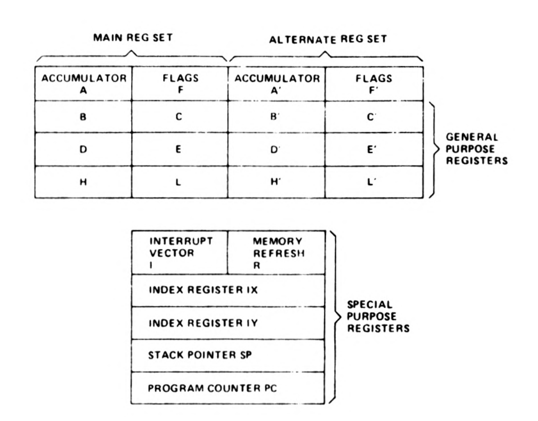


Fig. 1.3 Registros del Z80-CPU

A continuación examinaremos detenidamente los distintos registros de abajo hacia arriba.

Contador del programa:

(Program Counter - PC -) Después de cada reset (o sea también después de cada aplicación de la tensión de alimentación), el contenido de este registro será de 0000H. La CPU buscará cada instrucción que esté almacenada en la memoria de esta dirección e incrementará

(incrementar = aumentar en 1) el contador del programa. Después decodificará la instrucción y la ejecutará (en el caso de ser una instrucción de 1 Byte) o bien buscará el próximo Byte de la instrucción en la memoria. El contador del programa contiene siempre la dirección de 16 Bits de aquella posición de memoria, de la que se buscará el próximo Byte.

Apuntador de pila:

(Stack Pointer - SP -) En el caso de interrupts o bien subprogramas, a menudo se exige salvar el contenido de algunos e incluso de todos los registros (pares de registros) de la cantidad AF, BC, DE, HL, IX, IY, porque las rutinas modifican los contenidos de los registros. El Z80-CPU ofrece para ello instrucciones especiales (PUSH, POP) con las que se podrán depositar contenidos de pares de registros en la "columna" y desde aquí poderlos recargar en los pares de registros correspondientes (por ejemplo al final de la rutina). La "pila" no es otra cosa que un área reservada para las operaciones PUSH-POP de la memoria RAM, pudiéndose elegir libremente por el programador la posición de la pila (cargar el apuntador de la pila con la correspondiente dirección). El apuntador de pila señala siempre sobre la última entrada de la misma y aumenta nuevas anotaciones hacia direcciones menores. El concepto "pila" se apoya sobre el significado de la definición del lenguaje usual. En la pila se depositará siempre componente por componente y se buscará de la pila en cuestión en la sucesión inversa (LIFO-Last In First Out-Organisation). Las operaciones de la pila serán automáticamente organizadas por la CPU (incremento resp. disminución del apuntador de pila).

Registros de índices IX e IY:

Estos dos registros de 16 bits independientes son de especial importancia para el direccionamiento indexado. Contienen una dirección básica de 16 bits, a la que se le añadirá internamente un espacio de 8 bits, contenido en la instrucción, de acuerdo con la técnica de complemento binario. La suma de la dirección básica y el espacio dará por resultado la dirección de la memoria que contiene el operando.

Registro de interrupt (I):

Este registro sólo tiene importancia en la modalidad de interrupt 2 del Z80-CPU. Contiene los 8 bits superiores del vector de interrupt, mientras que los 8 bits inferiores son ofrecidos por el componente de interrupción. Ambas mitades de 8 bits son unidas por la CPC en un vector completo de 16 bits que señala hacia una tabla de salto de la memoria. La tabla de salto contiene las direcciones definitivas de las correspondientes rutinas de servicio. El contenido del registro del interrupt puede ser libremente programado, pudiéndose así localizar dinámicamente las rutinas de servicio (rutinas de interrupt) en el área total de la memoria.

Registro de refresh (R):

El registro de refresh contiene las direcciones de refresco para RAM's dinámicas, de modo que puede omitirse la lógica de refresco exigida adicionalmente para ello. A pesar de tratarse de un registro de 8 bits, al "refresh" únicamente le sirven los siete bits inferiores. Después de cada ciclo de vacío de instrucciones se incrementará el registro de refresh (módulo 128-contador), de modo que el refresco de las RAM,s

dinámicas se realiza de manera "transparente" para la CPU sin ralentizar ninguno de los procesos. El registro de refresh queda normalmente inutilizado por el programador y no influye de ningún modo en el proceso del programa.

Los registros universales:

El Z80-CPU posee dos juegos idénticos de registros universales, de los que únicamente puede utilizarse un sólo juego; mediante la instrucción EXX podrán realizarse entre los mismos las conexiones oportunas. Esta posibilidad es importante cuando se trata de rutinas muy rápidas de servicio de interrupt, ya que de este modo los registros universales no deberán ponerse primeramente en la pila. El primer juego de registros consta de los registros B, C, D, E, H y L y el segundo juego de los registros B', C', D', H' y L'. Estos registros podrán aplicarse ya sea individualmente (8 bits) o bien a pares (16 bits). Los siguientes pares de registros pueden formarse según sigue: BC, DE; HL, BC', DE' y HL'. En sistemas mínimos que únicamente están equipados con una memoria de valor fijo (ROM), podrá utilizarse el juego de registros binarios para la memorización intermedia de datos.

Acumulador y registro flag:

El Z80-µP tiene dos acumuladores (A y A') y dos registros flag (F y F'). Con la instrucción EX, AF, AF' podrá ser conmutado oportunamente el otro juego. El acumulador contiene los resultados de operaciones aritméticas y lógicas. El registro flag contiene informaciones sobre estados especiales de resultados (por ejemplo resultado cero, paridad impar del resultado, etc.).

1.5 Ejecución de instrucciones:

Después de habernos familiarizado con el concepto general del Hardware del Z80-CPU, veremos a continuación cómo se realiza la ejecución de instrucciones en el Z80. En principio todas las operaciones del Z80 y de los ordenadores convencionales (tanto si son ordenadores pequeños, grandes o de tamaño superior) pueden atribuirse a una fórmula banal que consta de tres pasos, a saber:

1.) Buscar instrucción (Fetch)
2.) Decodificar instrucción (Decode)
3.) Ejecutar instrucción (Execute)

No debería ser demasiado difícil suponer que este principio puede conllevar a una determinada rigidez en el establecimiento del programa, en el que algunos problemas difícilmente podrían solucionarse dentro de la técnica del mismo. Esto es sólo un comentario al margen.

Las tres fases de la ejecución de las instrucciones serán a continuación detenidamente examinadas.

1. Buscar instrucción:

El contador del programa del Z80-µP estará siempre cargado después de la conexión, respectivamente después de un reset, con la dirección 0000H. El procesador se buscará en primer lugar la primera instrucción de dicha dirección. Este es también el motivo por el que la memoria de valor fijo (ROM) del sistema se encuentra normalmente en el área inferior de la memoria.

La ROM contiene el programa necesario para la función deseada del sistema. Un programa consta de instrucciones. Una instrucción consta a su vez de varios Bytes. La Fig. 1.4 muestra el plan de ocupación de la memoria de un sistema típico. La Fig. 1.5 muestra un ejemplo para las dos primeras instrucciones de la ROM.

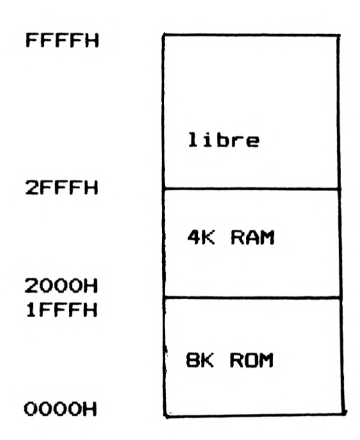


Fig. 1.4 Ocupación típica de la memoria

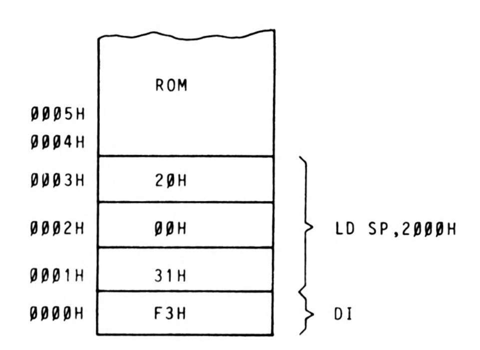


Fig. 1.5 Ejemplo para el almacenaje de las instrucciones DI y LD SP,2000H de la ROM.

Después de que la CPU haya depositado el contenido del contador del programa en el bus de dirección y de que haya activado para su lectura la memoria mediante las líneas de control RD y MREQ, la memoria dará el contenido de la posición de memoria direccionada sobre el bus de datos, desde donde será leído por la CPU. En nuestro caso se trataría de la fecha F3H, lo que corresponde a la instrucción del ensamblador DI. Cada vez que se lee un byte de instrucción se aumenta automáticamente el contenido del contador del programa en 1, de modo que el contador del programa direccionará siempre el próximo byte a ser leído. De este modo se origina un mecanismo automático de correlación de instrucciones.

2. Decodificar instrucciones:

La instrucción buscada de la memoria se decodificará

mediante un decodificador interno, o sea que será convertida en las acciones correspondientes.

3. Ejecutar instrucciones:

La ejecución de las instrucciones parte sigue después de la decodificación de las mismas. En nuestro ejemplo la ejecucion de éstas consiste en la postergación de los flip-flop de interrupt. La instrucción DI es una instrucción de un byte. Sin embargo muchas instrucciones, tal como lo demuestra nuestro siguiente ejemplo, están formadas por varios bytes. En estos casos el primer byte de información de forma codificada ("en clave") contiene la información sobre la cantidad de bytes que la CPU deberá leer todavía, para que se acepte por completo la instrucción al caso.

En principio sabemos lo que ocurre durante las tres fases. Sin embargo todos los procesos deberán estar a tiempo y exactamente sincronizados. La Fig. 1.6 muestra el proceso temporal de un ciclo típico de instrucciones.

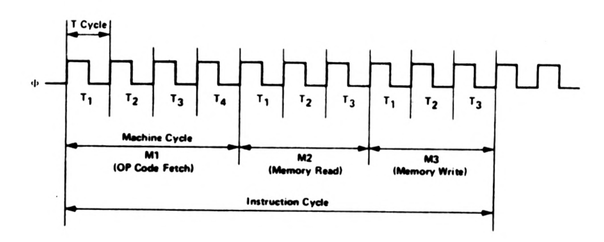


Fig. 1.6 Proceso típico de un ciclo de instrucciones

Aquí distinguiremos tres ciclos diferentes:

1.) Ciclo de cadencia (T-Cycle). Un ciclo de cadencia corresponde a la duración de los periodos de la cadencia del sistema 0. La cadencia del sistema sirve para la sincronización de todos los procesos dentro del sistema. Los ciclos de cadencia se denominan mediante T1, T2, etc.

2.) Los ciclos de la máquina (Machine Cycle). La CPU requiere para cada una de las fases de ejecución de las instrucciones buscar, decodificar y ejecutar por lo menos en tres ciclos de cadencia. Estos ciclos de cadencia son recopilados en el llamado ciclo de la máquina. La mayoría de las instrucciones necesitan varios ciclos de máquina.

3.) Ciclo de instrucción (Instrution Cycle). Un ciclo de instrucción comprende todos los ciclos de la máquina que se requieren para buscar, decodificar y ejecutar una instrucción.

Durante T1 se depositará siempre la dirección del próximo byte a ser buscado sobre el bus de dirección.

Durante T2 se realiza el incremento del contador del programa. Además, la memoria tiene tiempo de emitir los datos que serán leídos por la CPU durante el flanco de cadencia que va en aumento entre T2 y T3. Durante T3 y T4 la CPU decodifica la instrucción y la ejecuta. A veces se requiere otra cadencia.

Esto es una representación bastante sencilla de la ejecución de instrucciones, pero indica los procesos iniciales que serán suficientes para la mayoría de los

lectores, ya que un tratamiento más detallado del tema no aportaría una mayor utilidad.

1.6 Flags del Z80-CPU:

Cada uno de los dos registros flag de la CPU posee seis flags:

D7	D6	D5	D4	D3	D2	D1	D0
S	Z	-	H	-	P/V	N	C

Los flags se activarán o desactivarán mediante diversas operaciones de la CPU. El estado de los flags C, Z, S y P/V podrá ser examinado a través de varias instrucciones, es decir que la ejecución de estas instrucciones depende del estado de los flags correspondientes. La función de los distintos flags es la siguiente:

C (Carryflag, flag de transmisión) Este flag queda influenciado por todas las operaciones aritméticas. Siempre cuando no pueda representarse el resultado de una suma con 8 bits (suma anterior) o bien cuando en una resta se requiera un llevando (Borrow), se aplicará este flag. Casi todas las instrucciones de desplazamiento y de rotación influencian a este flag.

Z (Zero flag, flag cero) Siempre cuando el acumulador tenga un contenido cero debido al resultado de una

operación, se aplicará este flag. En la instrucción BIT tendrá el estado invertido del bit analizado.

Además está activado, cuando en las instrucciones de entrada y de salida del componente INI, IND, OUTI y OUTD se disminuye a cero el registro B (contador de bytes). En las instrucciones de búsqueda de componente indicará la igualdad de ambos operandos. Las instrucciones de entrada y de salida del componente INIR, INDR, OTIR y OTDR activan siempre el flag Z.

S (Signum, signo) Para procesar cifras con signos se utilizará en el Z80-µP la llamada técnica de complementos binarios. El bit 7 representa el signo de la cifra de 7 bits y está activado para números negativos (lógico 1). El flag S está activado, cuando el resultado de una operación sea negativo (D7 = lógico 1).

En la instrucción IN r,(C) se indica el estado del bit 7 de la fecha leída. Esto es especialmente importante para sistemas con técnica de consulta (Polling), porque el bit 7 indica el registro de estado de componentes periféricos, siendo generalmente el estado de "listo", y porque con la instrucción BIT no pueden examinarse los distintos bits de los registros externos de estado (con ello pueden examinarse únicamente bits en el área de memoria, pero no en el área de entrada/salida).

P/V (Parity/Overflow, paridad/desbordamiento) En todas las operaciones lógicas (AND, OR, XOR, RL, SRL, etc.) está activado este flag, cuando la parídad del resultado es par. Cuando se trata de paridad

este flag está desactivado. De esta manera podrá examinarse asímismo la paridad de una fecha leída en un componente periférico utilizando la instrucción IN r(C). Esto es útil para la transmisión de datos con el fin de poder detectar posibles fallos que hayan surgido durante la transmisión. Los signos normalmente se transmiten en el código ASCII de 7 bits. Añadiendo un bit de paridad podrá asegurarse dicho código contra fallos de transmisión, sin tener que utilizar otros bytes (7 bits ASCII + 1 bit de paridad = 8 bits = 1 byte = formato estándar).

En todas las instrucciones aritméticas este flag indica un desbordamiento, es decir, una suma anterior del bit 6 al bit 7. Este desbordamiento aparece siempre cuando el resultado sobrepasa o bien queda por debajo del área numérica para la técnica de complementos binarios (-128 hasta +127). Aquí deberá mencionarse que los procesadores 8080 y 8085 no poseen ningún flag con función de desbordamiento. Las instrucciones LD A,I y LD A,R efectúan la carga del interrupt del Flip-Flop IFF2 después del P/V. El contenido del IFF2 podrá así examinarse o bien salvarse.

En las instrucciones de transferencia de componentes LDI y LDD así como en las instrucciones de búsqueda de componentes CPI, CPIR, CPD y CPDR está activado el P/V, cuando el par de registros BC se disminuya hasta cero.

Los siguientes dos flags se necesitarán para la aritmética BCD y no podrán ser examinados:

H (Half Carry, media retención) Ya que cada cifra BCD está codificada con cuatro bits, un byte contiene un número BCD de dos cifras. Si en operaciones aritméticas tiene lugar una retención o bien un llevando de la cifra BCD inferior a la superior, es activado el flag H. La instrucción DAA (combinación decimal) lo utiliza para la adaptación del resultado al valor adecuado.

N (Resta) Este bit está activado después de cada resta y desactivado después de cada suma. Para el programador no es de importancia, ya que se valorará la CPU internamente en relación con la instrucción DAA (combinación decimal). La función de esta instrucción depende de si la operación anterior era una suma o bien una resta.

La tabla 1.7 ofrece una representación resumida de las operaciones de los flags.

Z 80 - CPU Hardware

Instruction	C	Z	P/V	S	N	H	Comments
ADD A, s; ADC A,s	↕	↕	V	↕	0	↕	8-bit add or add with carry
SUB s; SBC A, s, CP s, NEG	↕	↕	V	↕	1	↕	8-bit subtract, subtract with carry, compare and negate accumulator
AND s	0	↕	P	↕	0	1	Logical operations
OR s; XOR s	0	↕	P	↕	0	0	And set's different flags
INC s	•	↕	V	↕	0	↕	8-bit increment
DEC m	•	↕	V	↕	1	↕	8-bit decrement
ADD DD, ss	↕	•	•	•	0	X	16-bit add
ADC HL, ss	↕	↕	V	↕	0	X	16-bit add with carry
SBC HL, ss	↕	↕	V	↕	1	X	16-bit subtract with carry
RLA; RLCA, RRA, RRCA	↕	•	•	•	0	0	Rotate accumulator
RL m; RLC m; RR m; RRC m SLA m; SRA m; SRL m	↕	↕	P	↕	0	0	Rotate and shift location m
RLD, RRD	•	↕	P	↕	0	0	Rotate digit left and right
DAA	↕	↕	P	↕	•	↕	Decimal adjust accumulator
CPL	•	•	•	•	1	1	Complement accumulator
SCF	1	•	•	•	0	0	Set carry
CCF	↕	•	•	•	0	X	Complement carry
IN r, (C)	•	↕	P	↕	0	0	Input register indirect
INI; IND; OUTI; OUTD	•	↕	X	X	1	X	Block input and output
INIR; INDR; OTIR; OTDR	•	1	X	X	1	X	Z = 0 if B ≠ 0 otherwise Z = 1
LDI, LDD	•	X	↕	X	0	0	Block transfer instructions
LDIR, LDDR	•	X	0	X	0	0	P/V = 1 if BC ≠ 0, otherwise P/V = 0
CPI, CPIR, CPD, CPDR	•	↕	↕	↕	1	X	Block search instructions Z = 1 if A = (HL), otherwise Z = 0 P/V = 1 if BC ≠ 0, otherwise P/V = 0
LD A, I; LD A, R	•	↕	IFF	↕	0	0	The content of the interrupt enable flip-flop (IFF) is copied into the P/V flag
BIT b, s	•	↕	X	X	0	1	The state of bit b of location s is copied into the Z flag
NEG	↕	↕	V	↕	1	↕	Negate accumulator

Tabla 1.7 Cuadro sinóptico de todas las operaciones de los flags.

1.7 Comparación con el 8080/8085:

En el año 1971 Intel presentó en el mercado el microprocesador de 8 bits 8008. Este procesador elaborado según la técnica del PMOS tenía 48 instrucciones básicas y era, con un tiempo mínimo del ciclo de instrucciones de 12,5 µs, bastante lento. Sin embargo no dejó de ser el primer microprocesador de 8 bits. En 1974 le siguió el modelo 8080 con 72 instrucciones básicas y un tiempo mínimo del ciclo de instruciones de 1,5 µs. Para el funcionamiento del 8080 se requería además el componente generador de reloj/excitador 8224, así como el componente de control del sistema + el reloj excitador del bus de datos 8228. El 8080-µP formó la base para el microprocesador Z80 desarrollado por la firma Zilog; dicho procesador está disponible desde 1977, dispone de 158 instrucciones básicas y tiene un tiempo mínimo del ciclo de instrucciones de 1 µs. Paralelamente a esto continuó desarrollando Intel también su 8080 y ofrece asimismo el 8085 desde 1977. Está equipado con 74 instrucciones básicas y tiene un tiempo mínimo del ciclo de instrucciones de 0,8 µs. Para poder colocar las líneas de control adicionales del 8085 en la caja 40 pins, tuvieron que multiplexarse el bus de datos y la parte inferior del bus de dirección en líneas comunes. El demultiplexado es asumido, o bien por un componente especial (por ejemplo 74LS 373), o bien llevado internamente a cabo por componentes especiales periféricos (por ejemplo 8355 ROM y I/O).

Fig. 1.8 Comparación de las lineas de conexión mútuamente correspondientes de los procesadores 8080, 8085 y Z80.

	8Ø8Ø + 8228	8Ø85	Z8Ø
Direcciones	AØ - A15	ADØ - AD7 (ALE = log.1)	AØ - A15
Datos	DØ - D7	ADØ - AD7 (ALE = log.Ø)	DØ - D7
Mando	HLDA	HLDA	BUSAK
	HOLD	HOLD	BUSRQ
	Ø2	CLK	-
	INT	INTR	INT
	INTE	-	-
	WAIT	-	-
	READY	READY	WAIT
	RESET	RESET IN	RESET
	SYNC	-	H1
	INTA	INTA	M1 + IORQ
	MEMR	RD + IO/M	RD + MEMRQ
	MEMW	WR + IO/M	WR + MEMRQ
	I/O RD	RD + IO/M	RD + IORQ
	I/O WR	WR + IO/M	WR + IORQ
	BUSEN	-	-
	SSTB	-	-
	-	RST 5.5	-
	-	RST 6.5	-
	-	RST 7.5	-
	-	TRAP	NMI
	-	RESET OUT	-
	-	SØ,S1	-
	-	SID	-
	-	SOD	-
	-	ALE	-

	8Ø8Ø + 8228	8Ø85	Z8Ø
	-	-	RFSH
	-	-	HALT

La estructura de los registros de los procesadores 8080 y 8085 se diferencia en particular del Z80 en que únicamente poseen un juego de registros (A, F, B, C, D, E, H, L) (el Z80 tiene, como ya se sabe, dos juegos de registros).

Para una mejor disposición de la CPU 8085 y del Z80 se detallarán brevemente los desarrollos ulteriores más importantes frente a su predecesor, el 8080:

8080 — Z80

1.) No se exige ningún componente de control del sistema.
2.) La señal de cadencia TTL monofásica es suficiente (8080: Cadencia bifásica - generador propio de cadencia).
3.) Es suficiente una tensión de alimentación, es decir 5 V ± 5 % (8080: + 5V ± 5 %, -5 V ± 5 %, + 12 V ± 5 %).
4.) Refresco automático de RAM's dinámicas (por ésto es también muy adecuado para su aplicación en ordenadores personales y domésticos).
5.) Juego de instrucciones ampliado a 158 instrucciones básicas (8080: 72 instrucciones básicas), entre las que se encuentran las instrucciones prácticas de componente y de verificación.
6.) Dos registros de índices de 16 bits IX e IY para el direccionamiento indexado, así como un juego secundario de registros (A', F', B', C'; D', E', N', L').
7.) Dos modalidades adicionales de interrupt (IM1 e IM2).
8.) Una entrada adicional no mascarable del interrupt (NMI).

8080 - 8085

1.) No se exige ningún componente de control del sistema.
2.) No se exige ningún generador de cadencia (únicamente cuarzo, combinación LC o RC).
3.) Una tensión única de alimentación, o sea +5 V ± 5 %.
4.) Ampliación del conjunto de instrucciones por las instrucciones RIM y SIM (sirven entre otros para la entrada y salida de datos en serie).
5.) Cuatro entradas de interrupt adicionales (RST 5.5, RST 6.5, RST 7.5 y TRAP). Estos interrupts están vectorizados y tienen prioridad interna. El TRAP no es mascarable.
6.) Una salida de RESET para inicializar los demás componentes del sistema.
7.) Port en serie de entrada y salida de datos (SID: entrada serie de datos, SOD: salida en serie de datos). SID se leerá con la instrucción RIM. SOD es controlado con la instrucción SIM.

Debido a las diferencias entre el 8085 y el Z80 no existe entre ambos ninguna compatibilidad de software, a pesar de que los dos son compatibles hacia arriba con el 8080.

2. EL SOFTWARE DEL Z80-CPU

2.1 Generalidades

Este capítulo proporciona las bases elementales para la creación de programas para el Z80. Ya que el programar únicamente puede aprenderse practicando, el lector debería no sólo copiar lógicamente los programas indicados, sino que también debería intentar de modificarlos según sus propios conceptos. Con la comprensión conseguida de esta manera será posible crear programas más complejos y así solucionar problemas individuales con un sistema del Z80. La representación se refiere únicamente al llamado lenguaje de ensamblador del Z80 que prácticamente estará disponible en cualquier ordenador personal o doméstico. En este libro se utilizarán tan sólo aquellas instrucciones de ensamblador que correspondan directamente con las del Z80-CPU, porque las instrucciones puramente específicas de ensamblador serán parcialmente y de modo distinto comprendidas por diversos "ensambladores", tales como ORG, DEFT, GLOBAL, MACRO, etc. Un ensamblador es un programa que traduce automáticamente las distintas instrucciones de un programa ensamblador (Source Code) al código binario (formato de 8 bits, de 1 a 5 bytes) elaborado por el Z80 (Object Code). Un desensamblador es un programa que vuelve a traducir el Source Code al Object Code.

La ventaja especial del lenguaje ensamblador está por un lado en la programación similar al lenguaje máquina y por el otro en la rapidez de la elaboración de los programas. A ésto debemos añadirle el elevado tiempo dedicado para la programación, detalle que deberá

tenerse en cuenta al solucionar problemas para los que se dispone de poco tiempo. Lenguajes de programación de alto nivel, como son el BASIC, el FORTRAN o el PASCAL, ofrecen instrucciones de gran potencia que disminuyen radicalmente el tiempo de programación invertido, pero que son más lentas en la ejecución del programa y con las que no puede operarse como en el lenguaje máquina (por ejemplo la inicialización de un elemento periférico). En la práctica se utiliza a menudo una combinación de lenguaje de programación de alto nivel y de lenguaje ensamblador. Los lenguajes de programación de alto nivel no son específicos del Procesador, de modo que en este libro no serán profundamente tratados.

Al lector deberían serle familiares los conceptos básicos, tales como el sistema binario, el dual y el hexadecimal, así como la representación BCD y el código ASCII; de lo contrario deberíamos remitirnos a la gran oferta de literatura que se ofrece sobre la técnica de datos. A pesar de todo, estos conceptos (para refrescar viejos contenidos de la memoria) serán caracterizados nuevamente de forma abreviada:

Sistema binario:

Es todo sistema numérico basado en dos elementos.

Sistema decimal:

Es el sistema de numérico que se basa en 10 elementos (0....9). El valor de un número será ordenado en potencias de 10 (por ejemplo: $12 = 1 \cdot 10^1 + 2 \cdot 10^0$).

Sistema dual:

Es un caso especial del sistema binario en el que el valor de un número será totalmente ordenado según las potencias binarias.
(Ejemplo: 101B = $1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0$)

Sistema hexadecimal:

Es un sistema numérico que está basado en 16 elementos (0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F). El valor de un número será ordenado según las potencias.
(Ejemplo: 1AH = $1 \cdot 16^1 + 1 \cdot 16^0$)
A menudo no se indica el contenido de un byte (= 8 bits) en dual, sino que se realiza con dos caracteres hexadecimales.
(Ejemplo: 1000 0000B = 80H)
Para distinguir números decimales deberá añadirse a los números hexadecimales una "H".

Representación BCD

(Binary Coded Decimal) Es la codificación binaria de números decimales con 4 bits por cifra. (Por ejemplo: 13 = 0001 0011)

Código ASCII:

(American Standard of Information Interchange) Es la codificación binaria de 128 caracteres con 7 bits. (32 caracteres de control y 96 caracteres alfanuméricos)
Código internacionalmente utilizado. (Ejemplo: 01000001 = "A")

Los programas presentados trabajan únicamente con números enteros. El lector es remitido al capítulo 8

para la mejor comprensión de las distintas instrucciones.

2.2 Aritmética dual:

Números enteros positivos (0,1,2,3...)

Con 8 bits pueden formarse 2^8 = 256 combinaciones. Si le damos a los 8 bits (D7 -D0) valores de potencias binarias (2^7 - 2^0), podrá representarse de esta manera el campo de números de 0 - 255. Para la representación de números mayores en el sistema dual, deberá repartirse la cifra dual sobre varios bytes, ya que el Z80-CPU únicamente tiene un formato de datos de 8 bits.

Números enteros con signos antepuestos (...-3,-2,-1,0, 1,2,3...)

Para la representación y elaboración de cifras duales con signos antepuestos se aplicará la técnica del complemento binario. El bit de mayor valor define el signo. El 1 significa negativo y el 0 positivo. El complemento binario de un número dual se calcula complementando cada posición binaria individual añadiéndole al final el 1. Las cifras negativas se representan como complemento binario.

Ejemplos para complementos binarios:

+ 127	0111	1111
+ 2	0000	0010
+ 1	0000	0001
0	0000	0000
- 1	1111	1111
- 2	1111	1110
- 128	1000	0000

Una resta será reducida a una suma del complemento binario. En el caso de tener una suma anterior, ésta no se tendrá en cuenta. Lo importante es considerar la eventual aparición de un desbordamiento (Overflow). Este aparece cuando el campo numérico de -128 a +127 es sobrepasado por el resultado.

Todos los ejemplos siguientes sobre la aritmética dual funcionan tanto para números enteros positivos como para números con signos antepuestos (bit de mayor valor = signo, números negativos como complemento binario). En el primer caso deberá examinarse el flag C y en el segundo el flag P/V después de cada operación aritmética.

Suma de 8 bits

Campo numérico: 0 - 255 resp. -128 hasta +127.

Aquí disponemos de dos instrucciones: ADD y ADC. La instrucción ADD no tiene en cuenta el contenido del carryflag. Una suma anterior (carry) aparece siempre en las sumas, cuando el resultado ya no puede ser representado con 8 bits. El carryflag contiene entonces el noveno bit (valor 2^8).

Ejemplo: Al contenido del acumulador A deberá sumársele el contenido del registro B. La instrucción correspondiente será:

```
ADD A,B
```

Si el resultado es mayor que 255, automáticamente se activará el carryflag.

Este ejemplo ha sido muy sencillo porque ambos operandos estaban archivados en los registros del Z80-CPU. Muy a menudo se encuentran los operandos en la memoria. Por este motivo deberá tratarse este caso con la ayuda de un ejemplo.

Ejemplo: En la dirección de memoria 4000H se encuentra el operando 1 y en la dirección de memoria 5000H el operando 2. Ambos operandos deberán sumarse y el resultado se archivará en la dirección de memoria 6000H. La secuencia de instrucciones correspondiente será según sigue:

```
LD A,(4000H)    ; OP1 a A
LD HL,5000H     ; Dirección OP2 a HL
ADD A,(HL)      ; OP1 + OP2
LD (6000H)      ; Resultado a 6000H
```

En lugar de las direcciones 4000H, 5000H y 6000H puede aplicarse cualquier otra dirección. Mientras el ensamblador disponga de la directiva "ORG", podrá indicarse asímismo la dirección de forma simbólica, por ejemplo "ADR1".

Suma de 16 bits:

Campo numérico: 0 - 65535 resp. -32768 hasta +32767

Esta podrá realizarse con dos sumas correlativas de 8 bits. En primer lugar se suman los ocho bits inferiores de los operandos y se obtendrá la parte inferior del resultado. Después se sumarán los ocho bits superiores de los operandos junto con la suma anterior que se pueda tener (carry), de manera que se obtendrá la parte

superior del resultado. Si el resultado es mayor que 65535 (= 2^{16}- 1), entonces se aplicará el carry flag (podrá interpretarse como bit diecisieteavo con el valor 2^{16}, siempre y cuando se trabaje únicamente con números enteros positivos).

Ejemplo: Aquí partimos siempre de la base de que los operandos y el resultado están en la memoria.

Dirección de la memoria	Contenido de la memoria
4000H	Operando 1, bits D7 - D0
4001H	Operando 1, bits D15 - D8
5000H	Operando 2, bits D7 - D0
5001H	Operando 2, bits D15 - D8
6000H	Resultado , bits D7 - D0
6001H	Resultado , bits D15 - D8

El programa podría ser según sigue:

```
LD A,(4000H)        ; OP1 D7-D0 a A
LD HL,5000H         ; Dirección de OP2 D7-D0 a HL
ADD A,(HL)          ; Suma de los bytes inferiores
LD (6000H),A        ; Almacenar el resultado de la
                      mitad inferior
LD A,(4001H)        ; OP1 D15-D8 a A
INC HL              ; HL direcciona OP2 D15-D8
ADC A,(HL)          ; Suma de los bytes superiores + C
LD (6001H),A        ; Almacenar el resultado de la
                      mitad superior
```

Según este esquema podrán sumarse números duales con 24,32,40...bits (n.8 bits).

Suma de 24 bits:

Campo numérico: 0 - 16777215 resp. -8388608 hasta +8388607

Realizaremos esta suma según el esquema conocido, pero utilizaremos el direccionamiento indexado. Normalmente equivale a que un problema, por norma general, pueda ser solucionado de varias maneras aunque la eficiencia sea distinta.

Ejemplo: Operando 1 a partir de la dirección 4000H con valoración en aumento.
Operando 2 a partir de la dirección 5000H con valoración en aumento.
Resultado a partir de la dirección 6000H con valoración en aumento.

El programa podría ser según sigue:

```
LD IX,5000H       ; Dirección de OP2 D7-D0 a IX
LD A,(4000H)      ; OP1 D7-D0 a A
ADD A,(IX+0)      ; Suma sin suma anterior
LD (6000H),A      ; Almacenar resultado D7-D0
LD A, (4001H)     ; OP1 D15-D8 a A
ADC A,(IX+1)      ; Suma con suma anterior
LD A,(6001H)      ; Almacenar resultado D15-D8
LD A,(4002H)      ; OP1 D23-D16 a A
ADC A,(IX+2)      ; Suma con suma anterior
LD A,(6002H)      ; Almacenar resultado D23-D16
```

Suma de 16 bits con instrucciones de 16 bits:

Campo numérico: 0 - 65535 resp. -32768 hasta +32767

Los ejemplos anteriores utilizan el acumulador. El Z80-CPU permite asímismo de un modo más restringido la utilización del par de registros HL como acumulador.

Ejemplo: Operando 1 a partir de la dirección 4000H con valoración en aumento.
Operando 2 a partir de la dirección 5000H con valoración en aumento.
Resultado a partir de la dirección 6000H con valoración en aumento.

El programa podría ser según sigue:

```
LD HL,(4000H)     ; OP1 a HL (2 bytes)
LD DE,(5000H)     ; OP2 a DE (2 bytes)
ADD HL, DE        ; Suma sin suma anterior
LD (6000H),HL     ; Almacenar resultado (2 bytes)
```

Observe la efectividad de estas instrucciones especiales de 16 bits, con las que al mismo tiempo podrán cargarse, sumarse y almacenarse dos bytes.

Suma de 32 bits con instrucciones de 16 bits:

Campo numérico: 0 - 4.294.967.295 resp.
-2.147.483.648 hasta +2.147.483.647

Este formato en la práctica es suficiente para exigencias de precisión muy elevadas. Para la representación

de un número dual en este formato deberán prepararse cuatro bytes independientemente de la longitud de dicho número. Para poder ahorrar espacio de memoria se elegirá consecuentemente un formato que no sea mayor de lo requerido.

Ejemplo: Operando 1 a partir de la dirección 4000H con valoración en aumento.
Operando 2 a partir de la dirección 5000H con valoración en aumento.
Resultado a partir de la dirección 6000H con valoración en aumento.

El programa podría ser según sigue:

```
LD HL,(4000H)  ; Los 2 bytes inferiores del OP1 a HL
LD DE,(5000H)  ; Los 2 bytes inferiores del OP2 a DE
ADD HL,DE      ; Suma de 16 bits sin suma anterior
LD (6000H),HL  ; Almacenar el resultado de los 2
                 bytes inferiores
LD HL,(4002H)  ; Los 2 bytes superiores del OP1 a HL
LD DE,(5002H)  ; Los 2 bytes superiores del OP2 a DE
ADC HL,DE      ; Suma de 16 bits con suma anterior
LD (6002H),HL  ; Almacenar el resultado de los 2 bytes
                 superiores
```

Resta de 8 bits:

Campo numérico: 0 - 255 resp. -128 hasta +127

Análogamente como en la suma se tienen a disposición las instrucciones SUB y SBC. La resta dual funciona análogamente como la suma dual.

Ejemplo: Operando 1 sobre la dirección 4000H
Operando 2 sobre la dirección 5000H
Resultado sobre la dirección 6000H

El operando 2 deberá restarse del operando 1 y almacenarse en la dirección 6000H.

El programa podría ser según sigue:

```
LD A,(4000H)        ; OP1 a A
LD HL,(5000H)       ; Dirección del OP2 a HL
SUB A,(HL)          ; OP1 - OP2
LD (6000H),A        ; Resultado a 6000H
```

<u>Resta de 16 bits:</u>

Campo numérico: 0 - 65535 resp. -32768 hasta +32767

Análogamente con la suma de 16 bits, el programa podría ser como se indica a continuación:

```
LD A,(4000H)   ; OP1 inferior a A
LD HL,5000H    ; La dirección del OP2 inferior a HL
SUB A,(HL)     ; OP1 inferior - OP2 inferior sin suma
                 anterior
LD (6000H),A   ; Almacenar abajo el resultado
LD A,(4001H)   ; OP1 superior a A
INC HL         ; HL direcciona OP2 superior
SBC A,(HL)     ; OP1 superior - OP2 superior con suma
                 anterior
LD (6001H),A   ; Almacenar arriba el resultado
```

Resta de 24 bits:

Campo numérico: 0 - 16.777.215 resp. -8.388.608
hasta +8.388.607

Análogamente con la adición de 24 bits, el programa podría ser (como se indica a continuación):

```
LD A,(4000H)    ; OP1 D7-D0 a A
LD HL,5000H     ; La dirección de OP2 D7-D0 a HL
SUB A, (HL)     ; Resta sin suma anterior
LD (6000H),A    ; Almacenar resultado D7-D0
LD A,(4000H)    ; OP1 D15-D8 a A
INC HL          ; HL direcciona OP2 D15-D8
SBC A,(HL)      ; Resta con suma anterior
LD (6000H),A    ; Almacenar resultado D15-D8
```

Resta de 16 bits con instrucciones de 16 bits:

Campo numérico: 0 - 65.535 resp. -32.768 hasta +32.767

Aquí únicamente se tiene una instrucción de resta de 16 bits a disposición, es decir, la SBC. La instrucción de resta sin suma anterior SUB se formará mediante la secuencia de informaciones

```
AND A
SBC ...
```

En lugar de la instrucción AND A que pospone el carry flag, podrá aplicarse, por ejemplo, la instrucción OR A o bien XOR A.

Análogamente con la suma de 16 bits y con instrucciones de 16 bits, el programa podría ser según sigue:

```
LD HL,(4000H)      ; OP1 a HL (2 bytes)
LD DE,(5000H)      ; OP2 a DE (2 bytes)
AND A              ; Posponer el flag C
SBC HL,DE          ; OP1-OP2
LD (6000H),HL      ; Almacenar resultado (2 bytes)
```

<u>Resta de 32 bits con instrucciones de 16 bits:</u>

Campo numérico: 0 - 4.294.967.295 resp. -2.147.483.648 hasta +2.147.483.647

La instrucción de resta SUB se formará esta vez con la secuencia de instrucciones

```
XOR A
SBC ...
```

Análogamente con la resta de 32 bits y con instrucciones de 16 bits, el programa podría ser como sigue:

```
LD HL,(4000H) ; Los 2 bytes inferiores de OP1 a HL
LD DE,(5000H) ; Los 2 bytes inferiores de OP2 a DE
XOR A         ; Posponer el flag C
SBC HL,DE     ; Resta de 16 bits
LD (6000H),HL ; Almacenar el resultado de los 2 bytes
                inferiores
LD HL,(4002H) ; Los 2 bytes superiores de OP1 a HL
LD DE,(5002H) ; Los 2 bytes superiores de OP2 a DE
SBC HL,DE     ; Resta de 16 bits con suma anterior
LD (6002H),HL ; Almacenar el resultado de los 2 bytes
                superiores
```

Principios generales de la multiplicación dual:

Existen muchas posibilidades de realización en técnicas de programación dentro de una multiplicación dual. Ya que el Z80-µP únicamente puede sumar y restar (como operación aritmética) deberá atribuirse cada multiplicación a (varias) sumas. Los siguientes dos ejemplos ilustran los dos principios básicos de la multiplicación dual:

```
                MPD    MPR                            MPD    MPR
Ejemplo: a)     1101 x 1011        Ejemplo: b)        1101 x 1011

                1101                                     1101
                 0000                                   1101
                  1101                                 0000
                   1101                               1101
                ___________                           ___________

                10001111                              10001111
                ===========                           ===========
```

En el ejemplo a) el factor multiplicador es unido de izquierda a derecha con el factor multiplicando. Los resultados parciales son o bien iguales al multiplicando (MPD) o iguales a 0. Cada resultado parcial quedará una posición más hacia la derecha que el anterior. La suma de los resultados parciales desplazados es igual al producto de los factores multiplicando y multiplicador.

En el ejemplo b) el multiplicador es unido con el multiplicando en una secuencia invertida, es decir, de derecha a izquierda. Los resultados parciales se habrán desplazado correspondientemente hacia la izquierda y no

hacia la derecha. La suma de los resultados parciales da nuevamente por resultado el producto del multiplicando y del multiplicador.

Las multiplicaciones duales con el Z80-µP se realizan análogamente pero con una diferencia: Los resultados parciales se sumarán de inmediato formando el llamado "resultado intermedio". Suponiendo que el próximo resultado parcial es igual al factor multiplicando, en principio podremos desplazar ya sea el multiplicando relativamente al resultado intermedio o bien el resultado intermedio relativamente al multiplicando. Aunque el resultado parcial sea "0", el "desplazamiento" deberá efectuarse.

El multiplicando, el multiplicador y el producto intermedio se encuentran normalmente en los registros del Z80. El desplazamiento de los contenidos del registro se posibilita mediante instrucciones especiales de desplazamiento (únicamente para registros directos de 8 bits). Si se desea desplazar el contenido de un registro hacia la izquierda, ello también podrá conseguirse sumando los contenidos del registro entre sí mismos (por ejemplo ADD A,A). De este modo también puede desplazarse de una sola vez (por ejemplo ADD HL,HL) un par de registros (16 bits).

El multiplicador puede desplazarse hacia la izquierda o la derecha (según sea el principio) en el flag de suma anterior (carry). El contenido del carryflag determina si el siguiente resultado parcial es igual al multiplicando o bien igual a cero.

Resumiendo obtendremos los siguientes principios de la multiplicación dual:

1.) Desplazar el MPR hacia la izquierda en el flag C
    C=0: desplazar el ZE hacia la izquierda
    C=1: desplazar el ZE hacia la izquierda y sumarle el MPD.
    Repetir el proceso tantas veces como posiciones tenga el MPR.

2.) Desplazar el MPR hacia la izquierda en el flag C
    C=0: desplazar el MPD hacia la derecha
    C=1: desplazar el MPD hacia la derecha y sumarle el MPD.
    Repetir el proceso tantas veces como posiciones tenga el MPR.

3.) Desplazar el MPR hacia la derecha en el flag C
    C=0: desplazar el ZE hacia la derecha
    C=1: desplazar el ZE hacia la derecha y sumarle el MPD.
    Repetir el proceso tantas veces como posiciones tenga el MPR.

4.) Desplazar el MPR hacia la derecha en el flag C
    C=0: desplazar el MPD hacia la izquierda
    C=1: desplazar el MPD hacia la izquierda y sumarle el ZE.
    Repetir el proceso tantas veces como posiciones tenga el MPR.

<u>Multiplicación dual de 8 x 8 bits con resultado de 16 bits</u>

La dirección 4000H contiene el multiplicando.
La dirección 5000H contiene el multiplicador.
La dirección 6000H contiene el resultado (byte de menor valor).
La dirección 6001H contiene el resultado (byte de mayor valor).

El programa podría ser según sigue:

```
MULT 8 x 8: LD HL,(4FFFH)
            LD L,0         ; Multiplicador a H
            LD DE,(4000)
            LD D,0         ; Multiplicando a E
            LD B,8         ; Contador en 8
MULT:       ADD HL,HL      ; MPR + ZE  desplazar izq.
            JR NC,KMULT    ; Salto relativo cuando C = 0
            ADD HL,DE      ; Suma de 16 bits
KMULT:      DJNZ MULT      ; Salto relativo, B=B-1
            LD (6000H),HL  ; Almacenar resultado
                             (2 bytes)
            RET            ; Salto de retroceso
```

El programa modifica los registros B,D,E,H y L.

La ocupación de registros es la siguiente:

L contiene el multiplicador
E contiene el multiplicando
HL contiene el resultado al final de la rutina.

El principio del programa es el siguiente:

El multiplicador (registro H) y el producto intermedio (registro L) serán desplazados mediante la instrucción "ADD HL,HL" un espacio hacia la izquierda en el flag de suma anterior (carry) y se verificará el contenido del flag mencionado.

Si C=0, se desplazará el bit siguiente del multiplicador al flag de suma anterior y se verificará. Al mismo tiempo se desplazará también el resultado intermedio en una posición hacia la izquierda (ADD HL,HL), siendo así preparado para una suma eventual con el multiplicando (si C=1).

Si C=0, el resultado intermedio (desplazado una posición hacia la izquierda) y el multiplicando se sumarán (ADD HL,DE). Tenga presente que el multiplicador (registro H) no sea influenciado por la suma de 16 bits "ADD HL,DE", ya que el contenido del registro D es cero.

Multiplicación dual de 16 x 16 bits con resultado en 16 bits

BC contiene el multiplicando
DE contiene el multiplicador
HL contiene el resultado
A sirve como contador de bucles

El programa podría ser según sigue:

```
MULT 16x16/16: LD HL,0        ; HL=0
               LD A,16        ; A=16, contador de bucles
               SLA E          ; Desplazar 16 bits hacia
                                la izquierda del
               RL D           ; Multiplicador
               JR NC,KADD     ; Verificación suma
                                anterior, salto C=0
               ADD HL,HL      ; Desplazar hacia la
                                izquierda el resultado
                                intermedio
               ADD HL,BC      ; Resultado intermedio +
                                Multiplicando
               JR NEU         ; Salto
KADD:          ADD HL,HL      ; Desplazar hacia la
                                izquierda el resultado
                                intermedio
NEU:           DEC A          ; Decrementar el contador
                                de bucles
               JP NZ,MULT     ; Repetir por si A≠0
               RET            ; Retroceso al programa
                                principal
```

El programa modifica los registros A,B,C,D,E,H y L.
Si los operandos estuviesen en la memoria, podrían ser fácilmente transferidos mediante las instrucciones de carga de 16 bits.

El principio del programa es el siguiente:

El multiplicador se desplazará bit a bit hacia la izquierda en el flag de suma anterior (carry) (con la ayuda de la construcción auxiliar SLA E y RL D) y se verificará el flag de suma anterior. Si C=0, se desplazará el resultado intermedio en un bit hacia la izquierda (ADD HL,HL). Si C=1, el ZE será asímismo desplazado hacia la izquierda además del multiplicando que será sumado al resultado intermedio (ADD HL,BC).

Multiplicación dual de 16 x 16 bits con resultado en 32 bits

BC contiene el multiplicando
DE contiene el multiplicador
HL contiene el resultado de la mitad inferior
IX contiene el resultado de la mitad superior
A sirve como contador de bucles

El programa podría ser según sigue:

```
MULT 16x16/32: LD A,16      ; A = contador de bucles
               LD HL,0      ; HL = 0
               LD IX,0      ; IX = 0
MULT:          SLA E        ; Desplazar 16 bits hacia
                              la izquierda del
               RL D         ; Multiplicador
               JR NC,KADD   ; Verif. suma ant., salto C=0
               ADD IX,IX    ; Desplazar el resultado
                              interm. arriba a la izq.
               ADD HL,HL    ; Desplazar el resultado
                              interm. abajo a la izq.
```

```
                JR NC,UEB    ; Verif. salto C=0
                INC IX       ; Sumar 1a a IX
UEB:            ADD HL,BC    ; Resultado intermedio +
                               multiplicando
                JR NC,NEU    ; Verif. salto C=0
                INC IX       ; Sumar 1a a IX
                JR NEU       ; Salto
KADD:           ADD IX,IX    ; Desplazar el resultado
                               intermedio hacia la izq.
                JR NC,NEU    ; Verif. salto C=0
                INC IX       ; Sumar 1a a IX
NEU:            DEC A        ; Disminuir cont. de bucles
                JP NZ,MULT   ; Repetición en caso de A/0
                RET          ; Retroceso al prog. princip.
```

El programa modifica los registros A,B,C,D,H,L e IX.
Si los operandos se encuentran en la memoria, éstos podrán ser fácilmente transferidos mediante las instrucciones de carga de 16 bits.

El principio del programa es el siguiente:

El multiplicador se desplazará bit a bit hacia la izquierda en el flag (carry) (ya que el Z80 no dispone de instrucciones especiales de desplazamiento de 16 bits, se utiliza la construcción auxiliar SLA E y RL D) y se verificará el flag. Si C = 0, el resultado intermedio (HL + IX) se desplazará en un bit hacia la izquierda (ADD IX,IX /ADD HL,HL) y se transferirá hacia IX. Si C = 1, se desplazará asimismo el resultado intermedio hacia la izquierda y se sumará además el multiplicando (BC) al resultado O producto intermedio.

(BC se sumará a HL; si aparece una suma anterior se transferirá a IX mediante INC IX.)

En cada pasada de bucle se origina en la derecha del multiplicador un espacio libre. Sería extraordinario poder utilizar este espacio libre como mitad superior del resultado. Sin embargo, esto podría motivar errores, ya que en cada pasada de bucle pueden aparecer (1º desplazamiento hacia la izquierda del resultado intermedio, 2º suma del multiplicando con el resultado intermedio) y por cada pasada de bucle se origina un espacio libre por desplazar hacia la izquierda el multiplicador.

Principios generales de la división dual

La división dual funciona según un esquema parecido al de la multiplicación dual; se atribuye a (varias) restas.

Se resta uno tras otro el divisor de los bits de mayor valor de los dividendos. De aquí se obtiene el resultado intermedio. El cociente se incrementará al mismo tiempo en 1. Después de cada resta se utilizará el resultado intermedio en lugar del dividendo inicial. Si el resultado intermedio es negativo después de una resta, se volverá a crear el estado inicial mediante la suma del divisor y restando el cociente. A continuación se desplazarán en una posición hacia la izquierda el dividendo y el cociente y se repetirá el algoritmo.

División dual de 16/8 bits con resultado de 8 bits y resto de 8 bits

- D Contiene el divisor
- HL Contiene el dividendo (al final de la rutina cociente)
- B Sirve como contador de bucles

El programa podría ser según sigue:

```
DIV 16/8:   LD E,0       ; E = 0
            LD B,9       ; B = 9, contador de bucles
            JP BEG       ; Salto  a BEG
DIV:        ADD HL,HL    ; Despl. el dividendo a la izq.
BEG:        AND A        ; Posponer el carryflag
            SBC HL,DE    ; Dividendo - Divisor
            INC HL       ; Cociente + 1
            JP P,KADD    ; Salto, si resto positivo
            ADD HL,DE    ; Restablecer, si
            DEC HL       ; resto negativo
KADD:       DJNZ DIV     ; Repetir, si B ≠ 0
            RET          ; Salto de retroceso al
                           programa inicial
```

La aritmética dual trabaja, como hemos visto, con formatos fijos (formato bit de 8,16,24,32, etc.). Es relativamente fácil de aplicar y cumple por completo con la mayoría de exigencias técnicas y matemáticas. En aplicaciones comerciales no deberá aparecer una pérdida de exactitud debido a la limitación del formato; en estos casos se trabajará con la aritmética BCD que permite cualquier número de posiciones sin estar sujeta a un formato determinado (ahorro de espacio de memoria).

2.3 Aritmética BCD:

Para la representación de una número decimal (0 - 9) se requieren 4 bits. Con cuatro bits son posibles no sólo 10 sinó también 16 combinaciones. Para poder compensar los errores que puedan surgir, deberá aplicarse después de cada operación aritmética la instrucción DAA. El resultado será nuevamente decimal (sin embargo binariamente codificado). Cada byte puede comprender dos números decimales binariamente codificadas (representación BCD empaquetada).

Suma BCD de 8 bits:

Campo numérico: 0 - 99

Ejemplo:

```
LD A,10H       ; Número BCD "11" a A
ADD A,14H      ; Suma el número BCD "14"
DAA            ; Igualación decimal
LD (6000H),A   ; Almacenar resultado
```

Todos los programas mostrados para la aritmética dual son también aplicables para la aritmética BCD, cuando después de cada instrucción aritmética (ADD, ADC, SUB, SBC) se intercala la instrucción DAA (igualación decimal).

(La instrucción DAA es asímismo práctica para la indicación decimal de valores duales sobre aparatos indicadores.)

Para poder representar números BCD de cualquier tamaño, se requiere un conjunto que, por ejemplo, puede tener el siguiente aspecto:

El primer nibble (1 nibble = 1 medio byte = 4 bits)

contiene el número de números de las que consta el número BCD.

El segundo nibble indica en qué posición de la derecha (la mantisa) se encuentra la coma.

El tercer nibble concreta el signo (por ejemplo 0000 = "+", 0001 = "-").

Los nibbles siguientes representan los distintos números del BCD (valoración descendente).

Programas con aritmética completa dual, respectivamente BCD, pueden obtenerse en ROM (aprox. 2 K).

2.4 Técnica del subprograma:

Hemos podido apreciar que el Z80-CPU después de ser conectado y empezando con la dirección 0000H procesa instrucción por instrucción. Mediante instrucciones especiales es posible poder dirigir el proceso del programa. Instrucciones de salto permiten por ejemplo dictar la dirección de la próxima instrucción a ser procesada por la CPU. Cuando se trata de saltos de retroceso (la dirección de la próxima instrucción es menor que la dirección actual) se forman de esta manera los llamados "bucles". Por lo menos una de las instrucciones que se encuentran en este bucle es normalmente una instrucción de salto condicionada que al entrar una condición determinada ocasiona un salto fuera del bucle, ya que de lo contrario la CPU recorrería el bucle hasta el infinito. Este tipo de bucles ya los hemos conocido en los ejemplos anteriores hechos para la multiplicación y la división. Evidentemente los bucles pueden estar también anidados entre si. Los bucles se utilizan para

repetir partes del programa cuya dirección de llamada (salto de retroceso) ya está concretada.

Por el contrario, los subprogramas son partes del programa que pueden ser llamadas con cualquier dirección y que únicamente se procesarán una sóla vez por llamada (naturalmente ésto no modifica nada que los subprogramas también puedan tener bucles). Esto ahorra el trabajo de escribir repetidamente secuencias de instrucciones. Los subprogramas se proveerán siempre de un nombre (Label) en modalidad de ensamblador y en la primera línea del programa, para cerrarse con una instrucción de retroceso. La llamada de un subprograma desde cualquier posición al programa principal se efectúa con la instrucción "CALL...". Suponiendo que el subprograma tiene el nombre de "SUB-1", podrá ser llamado mediante la instrucción "CALL SUB-1". Un subprograma iniciado de un programa principal finalizará generalmente con la instrucción de retroceso "RET". Esta instrucción de retroceso ocasiona la continuación del programa principal a partir de la posición de la llamada del subprograma (para más detalles véase el capítulo 8). En cada llamada de subprograma, el contador de instrucciones se pondrá automáticamente sobre la pila. Si el subprograma modifica los contenidos del registro que más adelante pueden ser requeridos por el programa principal, los contenidos del registro deberán ser salvados en la pila al principio de un subprograma mediante las correspondientes instrucciones de PUSH. Al final del subprograma deberán buscarse nuevamente de la pila los registros al caso mediante las instrucciones pertinentes de POP, pero con una sucesión inversa (la pila es una estructura Last-In-First-Out). Es muy posible anidar subprogramas, es decir, que un subprograma llama a otro subprograma,

etc.

Tal como veremos en el capítulo 4, los interrupts del hardware ocasionan una interrupción del proceso de instrucciones en curso. Si aparece un interrupt y éste es aceptado, la CPU bifurcará automáticamente hacia la dirección específica del interrupt. A partir de esta dirección deberá estar la rutina de servicio de interrupt adecuada (lo que no es otra cosa que un subprograma). Estos subprogramas especiales pueden cerrarse con tres instrucciones de retroceso distintas:

1.) RET: Esta ocasiona un retroceso al programa principal, eventualmente los componentes periféricos del Z80 conectado no reconocen de la misma el final de la rutina de servicio.

2.) RETI: Los componentes periféricos del Z80 reconocen de la misma el final de la rutina de servicio. Con esta instrucción deberían cerrarse rutinas que están destinadas a los interrupts de la entrada INT del Z80-CPU. Ya que la CPU después de haber aceptado un interrupt mascarado (interrupt en la entrada INT) queda cerrado para otros interrupts mascarados, la aceptación de interrupts de la CPU deberá abrirse mediante la instrucción explícita "EI". De esta manera también se posibilita el anidado de las rutinas de interrupt.

3.) RETN: Mediante esta instrucción deberían concluirse rutinas que están subordinadas a los interrupts de la entrada "NMI". La instrucción li-

bera nuevamente interrupts mascarados siempre y cuando fuesen liberados antes de aparecer los interrupts no mascarados.

Además de la instrucción de llamada del subprograma "CALL..." existe otra instrucción de llamada especial, o sea, la "RST P" (restart en la dirección P). Esto es una instrucción de 1 byte y por lo tanto es muy rápida. Desventajosa es la limitación de las posibles direcciones de llamada P sobre los 8 siguientes valores:

00H	=	00
08H	=	08
10H	=	16
18H	=	24
20H	=	32
28H	=	40
30H	=	48
38H	=	56

La mayoría de los subprogramas elaboran datos. El subprograma deberá saber dónde se encuentran dichos datos (parámetros). Existen tres posibilidades:

1.) Entrada de parámetros en los registros:
El subprograma espera que los parámetros requeridos estén en los registros determinados.

2.) Entrada de parámetros en la memoria:
El subprograma espera que los parámetros requeridos estén en las posiciones de la memoria determinadas y archiva los datos para procesarlos en los registros correspondientes o bien almacena el

contenido de los registros en la memoria (por ejemplo los resultados).

3.) Traspaso de parámetros en la pila:

El subprograma sabe en qué sucesión están archivados los datos en la pila y los busca mediante instrucciones POP apropiadas en los registros correspondientes. Aquí debe mencionarse que dentro de un programa principal pueden utilizarse varias pilas, con lo cual a cada subprograma se le asignará su propia pila. En este caso, el subprograma deberá modificar adecuadamente, antes de las operaciones de la pila, el indicador SP de la misma para volver a crear al final el verdadero contenido de la pila. Cuando se trata de subprogramas anidados, es conveniente intercalar el antiguo contenido SP mediante una instrucción de intercambio (por ejemplo EX SP,IX), ya que el mismo es específico para subprogramas.

Los programas de mayor capacidad se dividen básicamente en subprogramas, porque de este modo quedan estructurados y cada uno de ellos puede ser así verificado.

2.5 Diversos subprogramas:

Verificación de un carácter

La rutina deberá averiguar si el carácter en la posición de memoria 8000H es igual a 0, 5 o bien 7.

El programa podría ser según sigue:

```
ZEIT        LD A,(8000H)     ; Buscar Carácter
            CP 00            ; Carácter = 0 ?
            JP Z,CERO        ; Salto a rutina "CERO"
            CP 05            ; Carácter = 5 ?
            JP Z,CINCO       ; Salto a rutina "CINCO"
            CP 07            ; Carácter = 7 ?
            JP Z,SIETE       ; Salto a rutina "SIETE"
            JP NICHTGE       ; Salto si el carácter es
                               distinto
```

Verificación de un campo de caracteres

La misión de la rutina es averiguar si el signo en la posición de memoria 8000H corresponde a una letra mayúscula del ASCII (A....Z).

El programa podría ser según sigue:

```
ZEIBT       LD A,(8000H)     ; Cargar carácter en Accu
            AND 7FH          ; Bit de paridad del
            CP 41H           ; ASCII "A" ?
            JR C,NO          ; Carácter  demasiado peque-
                               ño? Salto
            CP 5AH           ; ASCII "Z" ?
            JR NC,NO         ; Carácter  demasiado  gran-
                               de? Salto
            LD A, 80H        ; 80H en A = Mayúscula
NO:         RET              ; Retorno al prog. principal
```

Al final del subprograma el bit 7 del accu, indica si el carácter era una letra mayúscula ASCII.

Generación del bit de paridad

Este programa crea un bit de paridad par en la posición D7. La transmisión de parámetros se realiza en el acumulador.

```
PARITG:     AND 7FH         ; Borra el bit 7
            JP PE,SI        ; Paridad par ? Salto
            OR 80H          ; Fija bit de paridad
SI:         RET             ; Retorno al prog. principal
```

Con el programa siguiente se crea un bit de paridad impar en la posición D7:

```
PARITU:     AND 7FH         ; Borra el bit 7
            JP PO,SI        ; Paridad impar ? Salto
            OR 80H          ; Fija bit de paridad
SI:         RET             ; Retorno al prog. principal
```

Verificación del cálculo de sumas

Para ello se utilizan distintos algoritmos. El programa presente calcula la suma de verificación de 255 elementos empezando por la dirección hexadecimal 8000H, y los compara con la suma de verificación que se encuentra almacenada en la posición 256. Si ambas sumas son distintas, entonces se saltará a un nuevo subprograma. La verificación del cálculo de sumas es un método para reconocer los posibles errores.

```
PRUEFS:     LD B,255       ; B = 255
            XOR A          ; Borrar suma de verificación
SCHL:       XOR (HL)       ; Calcular suma de verificación
            INC HL         ; Direc. próximo elemento
            DEC B          ; Disminuir contador de bucles
            JR NZ,SCHL     ; Repetir si B ≠ 0
            LD C,(80FFH)   ; El elemento 256 a C
            CP C           ; Comparación de sumas de
                             verificación
            CALL Z,ERROR   ; Error? Salto al subprograma
            RET            ; Retorno al prog. principal
```

El programa modifica los registros A,B,C,H y L

<u>Transferencia de componentes</u>

El programa siguiente transfiere las 8K inferiores (por ejemplo sistema operativo ROM) en el campo RAM a partir de 8000H.

```
BLKTR:      LD HL,0H       ; Primer elemento de la tabla
                             fuente
            LD DE,8000H    ; Primer elemento de la tabla
                             objeto
            LD BC,2000H    ; Número de elementos
            LDIR           ; Transferencia de componentes
            RET            ; Retorno al prog. principal
```

El programa modifica los pares de registros HL,BC y DE.

2.6 Tipos de direccionamiento:

Generalidades:

La mayoría de las instrucciones Z80 trabajan con datos que se encuentran ya sea en los registros CPU, en la memoria o bien en los Ports de entrada/salida. Los tipos de direccionamiento determinan la manera según la cual la CPU deberá tomar los correspondientes datos. Lamentablemente no se dispone siempre de todos los tipos de direccionamiento para los distintos grupos de instrucciones, de modo que deberán tenerse en cuenta toda una serie de limitaciones.

Direccionamiento inmediato (Immediate)

A la parte operacional le seguirá inmediatamente una fecha de 1 byte:

```
Ejemplo: AND 3CH
         ADD A,4BH
```

Direccionamiento inmediato ampliado (Immediate Extended)

A la parte operacional le seguirá inmediatamente una fecha de 2 bytes:

```
Ejemplo: LD BC, 76C3H
         LD HL, 4389H
```

Direcionamiento modificado de la página cero (Modified Page Zero)

Mediante la instrucción de 1 byte RST podrá saltarse a una de las 8 posibles direcciones de la "página cero" del campo de memoria (para más detalles véase el capítulo 8). Bajo "página cero" se sobreentiende el campo de memoria desde 0000H hasta 00FFH. Algunos procesadores (serie 68...) disponen de un tipo especial de direccionamiento ("direccionamiento reducido") al que a la parte operacional le sigue un dato de direccionamiento de 1 Byte, pudiéndose así buscar rápidamente datos del campo de memoria 0000H hasta 00FFH. La instrucción RST del Z80 únicamente podrá buscar en ocho direcciones dentro de este campo. Por este motivo este tipo de direccionamiento se llama direccionamiento "modificado" de la página cero.

Ejemplo: RST 08H

Direccionamiento relativo

A la parte operacional le sigue un dato de 1 byte de desplazamiento. Este dato se sumará según la técnica complementaria dual al contenido del contador del programa +2, pudiéndose así realizar saltos de +129 hasta -126. Programas que únicamente trabajan con instrucciones de salto relativas son libremente desplazables.

Ejemplo: JR 1CH

Direccionamiento ampliado (Extended)

A la parte operacional le sigue una parte de direccionamiento de 2 bytes.

```
Ejemplo: JP 1000H
         LD A,(34CBH)
```

Direccionamiento indexado (Indexed)

A la parte operacional le sigue un dato de 1 byte de desplazamiento. Este dato se sumará según la técnica complementaria dual al contenido de uno de los dos registros de índices IX y IY. El resultado sirve de indicador sobre la posición de la memoria deseada (el contenido del registro de índices permanecerá intacto). Este tipo de direccionamiento es especialmente efectivo si se trabaja con tablas.

```
Ejemplo: LD H,(IY+4)
         BIT 4,(IX+13)
```

Direccionamiento de registros

Los operandos se encuentran en los registros de la CPU.

```
Ejemplo: RES S,E
         LD  C,L
```

Direccionamiento implícito (Implied)

La parte operacional ya contiene una aclaración sobre el lugar de un operando.

Ejemplo: CP B; El accu contiene el segundo operando.
OR C; El accu contiene el segundo operando.

En todas las operaciones aritméticas el acumulador está direccionado implícitamente como objetivo para el resultado.

Direccionamiento indirecto de registros (Register Indirect)

La parte operacional especifica uno de los pares de registros de la CPU de 16 bits como indicador en una posición de la memoria.

Ejemplo: XOR (HL)
LD C,(HL)
JP (IY)

Direccionamiento de bits (bit)

La parte operacional especifica el bit del operando que deberá ser verificado, activado o desactivado. El operando podrá ser direccionado como registro indexado o bien como registro indirecto indexado.

Ejemplo: SET 5,B
RES 7,(IX+40)

2.7 Grupos de instrucciones del Z80:

Generalidades:

Este capítulo deberá ayudar a conseguir una visión estructurada del conjunto de instrucciones del Z80. Para el novato en programación puede ser suficiente para crear programas sencillos; deberán tomarse únicamente un par de instrucciones típicas de los ocho grupos disponibles para poder trabajar y así conseguir la noción de los diversos tipos de instrucciones sobre los que se dispone. Para la creación de programas efectivos se exige sin embargo un conocimiento exacto de todas las instrucciones del Z80. Una descripción detallada de todas las instrucciones podrá encontrarla en el capítulo 8 ("Conjunto completo de instrucciones del Z80"). Las tablas de este capítulo demuestran de inmediato todas las posibilidades del Z80 y las operaciones de instrucción del 8080, caracterizando los campos en gris cuyas instrucciones están disponibles en el 8080-µP.

Los campos contienen el código de la máquina de la instrucción correspondiente. (Tenga presente que a pesar de la identidad del código de máquina del 8080 con los correspondientes códigos de Z80, la escritura del ensamblador es distinta.)

Instrucciones de transferencia y de cambio:

Las instrucciones de transferencia sirven para el transporte de contenidos de células del registro y de la memoria. En el Z80 se distinguen cuatro grupos distintos de instrucciones de transferencia:

* Instrucciones de transferencia de 8 bits
* Instrucciones de transferencia de 16 bits
* Instrucciones de aplilado
* Instrucciones de transferencia de pilas

Instrucciones de transferencia de 8 bits:

Estas instrucciones sirven para el transporte de un byte (= 8 bits).

La escritura del ensamblador (nemotécnico) tiene el siguiente formato:

LD Objeto, fuente

La instrucción LD A,C efectúa la carga del contenido del registro C en el acumulador. El contenido de la fuente (registro C) queda intacto. Verbalmente se ofrece la forma "carga 'objeto' de la 'fuente'" para la formulación de las instrucciones de transferencia.

Instrucciones de transferencia de 16 bits:

Estas instrucciones transportan dos bytes, por ejemplo el contenido del par de registros IX en el par de registros SP (en la instrucción LD SP,IX). Sin embargo se puede también escribir con una sola instrucción dos bytes en la memoria o bien leerlos de la misma. Un ejemplo para ello sería la instrucción LD DE,(3840H) - (carga el par de registros DE de la dirección hexadecimal de la memoria 3840H). Como ya se sabe, las memorias de los sistemas del Z80 están orientadas en forma de bytes, es decir, que por dirección de memoria comprenden únicamente 8 bits. Por este motivo las instrucciones de transferencia de la memoria de 16 bits se refieren

siempre a dos posiciones de memoria consecutivas, de las que, sin embargo, por motivos de simplicidad, se indica únicamente la instrucción más baja.

Suponiendo que la memoria contiene en la dirección 3840H el dato 01H y en la dirección 3841H el dato 02H, el par de registros DE tendría después de la ejecución de la instrucción el contenido 0201H. Análogamente funciona también el almacenado, respectivamente carga, de los contenidos de pares de registros en relación con las operaciones de pila (PUSH-POP). El par de registros SP indica siempre la última entrada en la pila de la memoria. Si por ejemplo el par de registros AF de la pila es desplazado, disminuirá en primer lugar el indicador de la pila (SP) (ya que anteriormente indicaba la última entrada) y el registro A será almacenado en esta dirección. El indicador de la pila disminuirá nuevamente y se almacenará el registro F. Los procesos repetidos de disminución y de almacenaje se realizarán automáticamente durante la ejecución de la instrucción por el Z80-µP. Para todas las instrucciones de transferencia de la memoria de 16 bits sirve el principio de almacenar la parte de menor valor de la palabra de 16 bits en la dirección de memoria más baja y la parte de mayor valor en la más alta. Con las dos instrucciones EX AF,AF' y EXX pueden intercambiarse el conjunto de registros principales y secundarios del Z80-µP. Distintamente a las instrucciones de transferencia, no se pierde ningún contenido de registros al efectuarse las transcripciones. Ejecutando dos veces estas instrucciones podrá volverse a obtener el estado inicial.

Transferencia de componentes e instrucciones de búsqueda de componentes:

Con estas instrucciones podrán crearse hasta 65536 posiciones de memoria. En las instrucciones de transferencia de componentes, el par de registros HL indica la fuente de datos y el par de registros DE el destino de datos. El par de registros BC sirve de contador de bytes, o sea que indica el número de transferencias a ser realizadas. Las instrucciones LDI y LDR realizan por cada llamada únicamente una transferencia, pero preparan todo lo necesario para la siguiente transferencia a ser efectuada (disminución de BC, aumento respectivamente disminución del par de registros HL y DE). Por el contrario, las instrucciones LDIR y LDDR realizan en cada llamada la transferencia de un pila entero. Mediante la instrucción LDIR podría desplazarse, por ejemplo, un pila de 512 bytes empezando por la dirección 1000H y por el campo a partir de 4000H, si se indican los siguientes contenidos de registros:

(HL) = 1000H
(DE) = 4000H
(BC) = 200H = 512D

La pila desde 1000H hasta 1200H estará después de la ejecución de la instrucción en el campo 4000H hasta 4200H.

Posibilidades similares se ofrecen en las instrucciones de búsqueda de componentes. Estas comparan sistemáticamente el contenido de la posición de la memoria direccionada indirectamente por HL con el contenido del acumulador. BC sirve nuevamente como contador de bytes, pudiéndose investigar el campo completo de la memoria (65536 bytes) mediante un carácter determinado y con una única instrucción.

Instrucciones aritméticas y lógicas:

Para la aritmética y lógica de 8 bits están a disposición, con una sola excepción, todos los tipos de direccionamiento. Esta excepción es el aumento, respectivamente disminución, de un dato n direccionado. La meta de todas estas operaciones es siempre el acumulador.

El Z80-µP posee cuatro tipos aritméticos de instrucciones:

ADD - Suma sin acarreo
ADC - Suma con acarreo
SUB - Resta sin acarreo
SBC - Resta con acarreo

Además, existen tres instrucciones especiales: CPL, NEG y DAA. CPL forma el complemento de uno del acumulador (complementación de cada una de los acumuladores) y NEC crea el complemento de dos (complemento de uno más 1). DAA sirve para el ajuste decimal de la aritmética BCD.

Para la aritmética de 16 bits están preparadas, de modo restringido, las instrucciones ADD, ADC y SBC. No existe ninguna instrucción de resta de 16 bits que no tenga en cuenta el carryflag (acarreo). Dado el caso deberá posponerse antes de la resta - con la instrucción AND A.

Todas las instrucciones lógicas (AND,OR,EXOR,CP) trabajan únicamente con datos de 8 bits, donde los operandos serán entrelazados en forma de bit y el resultado almacenado en el acumulador. A continuación serán descritos más detalladamente:

AND: La tabla de la verdad de esta operación es:

```
0 AND 0 = 0
0 AND 1 = 0
1 AND 0 = 0
1 AND 1 = 1
```

La tabla de la verdad indica que el resultado de un enlac lógico AND será únicamente 1, si ambos operandos son 1.

(Si únicamente un operando es 0, el resultado será siempre 0.) Preferentemente se utiliza esta instrucción con el fin de volver a aplicar con exactitud determinados bits en el acumulador.

```
Ejemplo: 1.) (A) = 1 1 0 1   0 1 1 1 B
         2.) (D) = 1 1 1 1   1 1 1 0 B
         3.) Instrucción: AND D
         4.) (A) = 1 1 0 1   0 1 1 0 B
```

OR: La tabla de la verdad de esta operación es:

```
0 OR 0 = 0
0 OR 1 = 1
1 OR 0 = 1
1 OR 1 = 1
```

La tabla de la verdad indica que el resultado de un enlace lógico OR es únicamente 1 si por lo menos un operando es 1.

Unicamente cuando ambos operandos son 0, el resultado será asímismo 0. Esta instrucción normalmente se aplica para fijar bits determinados en el acumulador.

```
Ejemplo: 1.) (A) = 1 0 0 0  0 0 0 0 B
         2.) (E) = 0 1 0 1  0 1 0 1 B
         3.) Instrucción: OR E
         4.) (A) = 1 1 0 1  0 1 0 1 B
```

XOR: OR exclusivo. La tabla de la verdad de esta operación es:

0 XOR 0 = 0
0 XOR 1 = 1
1 XOR 0 = 1
1 XOR 1 = 0

La tabla de la verdad indica que el resultado de un enlace lógico exclusivo OR será únicamente 1, si únicamente un solo operando es 1 (si los dos operandos son iguales el resultado es 0). Aquí se ofrecen dos campos de aplicación para esta instrucción:

1.) El enlace "XOR" de dos bytes es exactamente cero cuando ambos bytes son idénticos. Este hecho puede aprovecharse para comparaciones.

2.) Cada 1 en el operando fuente efectúa un complemento de la posición correspondiente en el acumulador. De esta manera pueden complementarse el acumulador o partes del mismo.

Instrucciones de rotación y de desplazamiento:

Estas instrucciones pueden aplicarse tanto sobre contenidos de registro como sobre contenidos de memoria. En este caso se desplazan siempre contenidos. Las operaciones de rotación se caracterizan por el proceso de desplazamiento que se efectúa en un bucle cerrado, mientras que en las operaciones de desplazamiento los contenidos son desplazados en el carryflag. Según sea puesto en un bucle el carryflag durante las operaciones de rotación (RL,RR) o no (RLC,RRC), resultará una rotación de 9 bits, respectivamente 8 bits. En las instrucciones de desplazamiento SLA y SRL se añadirá un 0 en la posición que quede libre. La instrucción SRA forma una excepción: Para poder mantener el bit del signo (D7), no se someterá al proceso de desplazamiento. Tenga presente que es lo mismo el desplazamiento hacia la derecha en una posición de una división por 2 y el desplazamiento hacia la izquierda en una posición de una multiplicación por 2. En todas las instrucciones de rotación y de desplazamiento el carryflag tendrá el contenido del bit "sobrante". Las instrucciones RLD y RRD son instrucciones especiales que facilitan el cálculo con números BCD.

Instrucciones de manipulación de bit:

Con las mismas será posible poder activar, desactivar o verificar cualquier bit en un registro A,B,C,D,H o L, respectivamente en una de las 65536 posiciones de la memoria.

Instrucciones de bifurcación y de salto:

El proceso secuencial de la serie de instrucciones de un programa puede ser interrumpido mediante instrucciones de bifurcación y de salto, cargando el contador de instrucciones PC con la dirección objeto. Los saltos dentro del programa principal (JUMP) se denominan "saltos", mientras que bajo la denominación "bifurcación" se sobreentiende el salto a una subrutina (CALL), respectivamente el salto de retroceso (RETURN). Saltos incondicionales (unconditioned) se realizan siempre; los condicionales se efectúan únicamente cuando la condición que contiene la instrucción se ha realizado. Como condición puede servir el estado de uno de los flags Z,C,P/V y S. El Z80-µP posee la potente instrucción de salto DJNZ e (Decrement B and Jump relativ if B is non-zero). Posibilita saltos relativos en e (+127/-128), disminuye automáticamente el registro B, mientras éste no sea 0, y se presta especialmente como final de bucles. Tal como las instrucciones de salto, las instrucciones de bifurcación CALL y RET (return) pueden ser condicionales e incondicionales. Las rutinas de servicio del interrupt se concluyen normalmente mediante las instrucciones especiales de retroceso RETI o bien RETN, ya que únicamente con éstas pueden desactivarse automáticamente las demandas del interrupt de los componentes periféricos especiales del Z80.

La instrucción de bifurcación más rápida es la RST (restart). Esta permite un salto a una de las 8 direcciones de la parte inferior de la memoria (0H,8H,10H,-18H,20H,28H,30H y 38H).

Instrucciones de entrada/salida:

Las distintas técnicas de entrada/salida están descritas más detalladamente en el capítulo 4. Las instrucciones de entrada/salida están construidas de cara a la velocidad, utilizan únicamente una dirección de 8 bits y les bastan dos bytes de instrucción. Debido a la dirección de 8 bits pueden direccionarse como máximo 256 dispositivos de entrada/salida. Las instrucciones elementales son IN y OUT. El direccionamiento del dispositivo periférico puede realizarse ya sea absolutamente (por ejemplo IN A,(18H)) o bien indirectamente al registro (por ejemplo OUT (C), B). La ventaja del direccionamiento indirecto estriba en la manipulación más flexible del boftware impulsor. Si por ejemplo se modifica la dirección de un dispositivo periférico, entonces deberá modificarse únicamente el contenido del registro C. Además, esta instrucción permite el acceso a los registros A,B,C,D,E,H y L.

Las instrucciones de entrada/salida de componentes son de una elevada utilidad. Estas permiten la transferencia de hasta 256 bytes entre los dispositivos periféricos y la memoria. Su función corresponde a la de las instrucciones de transferencia de componentes (que únicamente son adecuadas para transferencias de memoria).

Instrucciones de mando del Z80-CPU:

Las instrucciones de mando sirven para la influencia en el tipo de funcionamiento de la CPU. El Z80-CPU conoce siete de estas instrucciones, cuya función será mencionada brevemente:

NOP: Durante una instrucción NOP la CPU no actúa, a no ser que las direcciones de refresco para las

memorias dinámicas sean emitidas y así se incremente el contador de instrucciones. Los NOPS se aplican por ejemplo para rellenar líneas de programa que se anulan o bien para reservar espacio para instrucciones que eventualmente tengan que añadirse.

HALT: Esta instrucción induce a la CPU a esperarse hasta que llegue un interrupt o bien un reset.

DI: (Disable Interrupt) Esta instrucción desactiva a los flip-flops del interrupt IFF1 e IFF2, cerrando así la entrada a los interrupts que van llegando.

EI: (Enable Interrupt) Esta instrucción activa el IFF1 y el IFF2, liberando de este modo los interrupts que van llegando.

IM 0: (Interrupt Modus 0) El componente periférico puede fijar una instrucción CALL o bien RST sobre el bus de datos (o sea una respuesta a una confirmación de interrupt).

IM 1: El procesador bifurca, en el caso de un interrupt, a la dirección 0038H.

IM 2: Es la modalidad más potente del Z80-µP. La dirección de inicio de la rutina de servicio de interrupt se encuentra en la tabla de saltos sobre la que señala el vector del interrupt. El vector del interrupt se forma del registro I del Z80-CPU y del vector I del componente interrumpido, representando el registro I la mitad superior del vector.

SOURCE

DESTINATION		IMPLIED		REGISTER							REG INDIRECT			INDEXED		EXT. ADDR.	IMME.
		I	R	A	B	C	D	E	H	L	(HL)	(BC)	(DE)	(IX + d)	(IY + d)	(nn)	n
REGISTER	A	ED 57	ED 5F	7F	78	79	7A	7B	7C	7D	7E	0A	1A	DD 7E d	FD 7E d	3A n n	3E n
	B			47	40	41	42	43	44	45	46			DD 46 d	FD 46 d		06 n
	C			4F	48	49	4A	4B	4C	4D	4E			DD 4E d	FD 4E d		0E n
	D			57	50	51	52	53	54	55	56			DD 56 d	FD 56 d		16 n
	E			5F	58	59	5A	5B	5C	5D	5E			DD 5E d	FD 5E d		1E n
	H			67	60	61	62	63	64	65	66			DD 66 d	FD 66 d		26 n
	L			6F	68	69	6A	6B	6C	6D	6E			DD 6E d	FD 6E d		2E n
REG INDIRECT	(HL)			77	70	71	72	73	74	75							36 n
	(BC)			02													
	(DE)			12													
INDEXED	(IX+d)			DD 77 d	DD 70 d	DD 71 d	DD 72 d	DD 73 d	DD 74 d	DD 75 d							DD 36 d n
	(IY+d)			FD 77 d	FD 70 d	FD 71 d	FD 72 d	FD 73 d	FD 74 d	FD 75 d							FD 36 d n
EXT ADDR	(nn)			32 n n													
IMPLIED	I			ED 47													
	R			ED 4F													

Fig. 2.2 Instrucciones de carga de 8 bits

		SOURCE									
		REGISTER							IMM. EXT.	EXT. ADDR.	REG. INDIR.
		AF	BC	DE	HL	SP	IX	IY	nn	(nn)	(SP)
DESTINATION REGISTER	AF										F1
	BC								01 n n	ED 4B n n	C1
	DE								11 n n	ED 5B n n	D1
	HL								21 n n	2A n n	E1
	SP				F9		DD F9	FD F9	31 n n	ED 7B n n	
	IX								DD 21 n n	DD 2A n n	DD E1
	IY								FD 21 n n	FD 2A n n	FD E1
EXT. ADDR.	(nn)		ED 43 n n	ED 53 n n	22 n n	ED 73 n n	DD 22 n n	FD 22 n n			
PUSH INSTRUCTIONS → REG. IND.	(SP)	F5	C5	D5	E5		DD E5	FD E5			

NOTE: The Push & Pop Instructions adjust the SP after every execution

↑ POP INSTRUCTIONS

Fig. 2.3 Instrucciones de carga de 16 bits "LD", "PUSH", "POP"

		IMPLIED ADDRESSING				
		AF'	BC', DE' & HL'	HL	IX	IY
IMPLIED	AF	08				
	BC, DE & HL		D9			
	DE			EB		
REG. INDIR.	(SP)			E3	DD E3	FD E3

Fig. 2.4 Instrucciones de intercambio "EX" y "EXX"

			SOURCE: REG. INDIR. (HL)	
DESTINATION	REG. INDIR.	(DE)	ED A0	'LDI' – Load (DE)←(HL) Inc HL & DE, Dec BC
			ED B0	'LDIR,' – Load (DE)←(HL) Inc HL & DE, Dec BC, Repeat until BC = 0
			ED A8	'LDD' – Load (DE)←(HL) Dec HL & DE, Dec BC
			ED B8	'LDDR' - Load (DE)←(HL) Dec HL & DE, Dec BC, Repeat until BC = 0

Reg HL points to source
Reg DE points to destination
Reg BC is byte counter

Fig. 2.5 Instrucciones de transferencia de componentes

SEARCH LOCATION: REG. INDIR. (HL)	
ED A1	'CPI' Inc HL, Dec BC
ED B1	'CPIR', Inc HL, Dec BC repeat until BC = 0 or find match
ED A9	'CPD' Dec HL & BC
ED B9	'CPDR' Dec HL & BC Repeat until BC = 0 or find match

HL points to location in memory
to be compared with accumulator
contents
BC is byte counter

Fig. 2.6 Instrucciones de búsqueda de componentes

SOURCE

	REGISTER ADDRESSING							REG. INDIR.	INDEXED		IMMED.
	A	B	C	D	E	H	L	(HL)	(IX+d)	(IY+d)	n
'ADD'	87	80	81	82	83	84	85	86	DD 86 d	FD 86 d	C6 n
ADD w CARRY 'ADC'	8F	88	89	8A	8B	8C	8D	8E	DD 8E d	FD 8E d	CE n
SUBTRACT 'SUB'	97	90	91	92	93	94	95	96	DD 96 d	FD 96 d	D6 n
SUB w CARRY 'SBC'	9F	98	99	9A	9B	9C	9D	9E	DD 9E d	FD 9E d	DE n
'AND'	A7	A0	A1	A2	A3	A4	A5	A6	DD A6 d	FD A6 d	E6 n
'XOR'	AF	A8	A9	AA	AB	AC	AD	AE	DD AE d	FD AE d	EE n
'OR'	B7	B0	B1	B2	B3	B4	B5	B6	DD B6 d	FD B6 d	F6 n
COMPARE 'CP'	BF	B8	B9	BA	BB	BC	BD	BE	DD BE d	FD BE d	FE n
INCREMENT 'INC'	3C	04	0C	14	1C	24	2C	34	DD 34 d	FD 34 d	
DECREMENT 'DEC'	3D	05	0D	15	1D	25	2D	35	DD 35 d	FD 35 d	

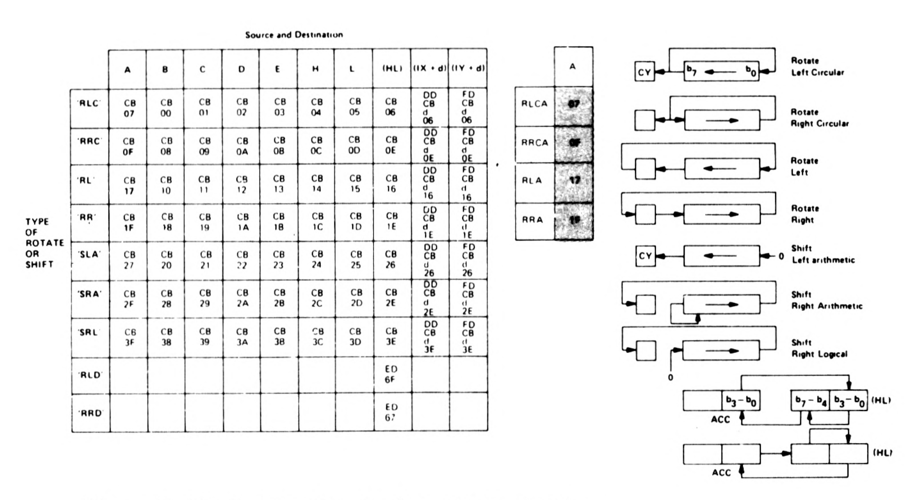
Fig. 2.7 Aritmética y lógica de 8 bits

Decimal Adjust Acc, 'DAA'	27
Complement Acc, 'CPL'	2F
Negate Acc, 'NEG' (2's complement)	ED 44
Complement Carry Flag, 'CCF'	3F
Set Carry Flag, 'SCF'	37

Fig. 2.8 Operaciones generales con AF

		SOURCE					
		BC	DE	HL	SP	IX	IY
DESTINATION 'ADD'	HL	09	19	29	39		
	IX	DD 09	DD 19		DD 39	DD 29	
	IY	FD 09	FD 19		FD 39		FD 29
ADD WITH CARRY AND SET FLAGS 'ADC'	HL	ED 4A	ED 5A	ED 6A	ED 7A		
SUB WITH CARRY AND SET FLAGS 'SBC'	HL	ED 42	ED 52	ED 62	ED 72		
INCREMENT 'INC'		03	13	23	33	DD 23	FD 23
DECREMENT 'DEC'		0B	1B	2B	3B	DD 2B	FD 2B

Fig. 2.9 Aritmética de 16 bits



Source and Destination

TYPE OF ROTATE OR SHIFT	A	B	C	D	E	H	L	(HL)	(IX + d)	(IY + d)
'RLC'	CB 07	CB 00	CB 01	CB 02	CB 03	CB 04	CB 05	CB 06	DD CB d 06	FD CB d 06
'RRC'	CB 0F	CB 08	CB 09	CB 0A	CB 0B	CB 0C	CB 0D	CB 0E	DD CB d 0E	FD CB d 0E
'RL'	CB 17	CB 10	CB 11	CB 12	CB 13	CB 14	CB 15	CB 16	DD CB d 16	FD CB d 16
'RR'	CB 1F	CB 18	CB 19	CB 1A	CB 1B	CB 1C	CB 1D	CB 1E	DD CB d 1E	FD CB d 1E
'SLA'	CB 27	CB 20	CB 21	CB 22	CB 23	CB 24	CB 25	CB 26	DD CB d 26	FD CB d 26
'SRA'	CB 2F	CB 28	CB 29	CB 2A	CB 2B	CB 2C	CB 2D	CB 2E	DD CB d 2E	FD CB d 2E
'SRL'	CB 3F	CB 38	CB 39	CB 3A	CB 3B	CB 3C	CB 3D	CB 3E	DD CB d 3E	FD CB d 3E
'RLD'								ED 6F		
'RRD'								ED 67		

	A
RLCA	07
RRCA	0F
RLA	17
RRA	1F

Fig. 2.10 Rotación y desplazamiento

		REGISTER ADDRESSING							REG. INDIR.	INDEXED	
	BIT	A	B	C	D	E	H	L	(HL)	(IX+d)	(IY+d)
TEST 'BIT'	0	CB 47	CB 40	CB 41	CB 42	CB 43	CB 44	CB 45	CB 46	DD CB d 46	FD CB d 46
	1	CB 4F	CB 48	CB 49	CB 4A	CB 4B	CB 4C	CB 4D	CB 4E	DD CB d 4E	FD CB d 4E
	2	CB 57	CB 50	CB 51	CB 52	CB 53	CB 54	CB 55	CB 56	DD CB d 56	FD CB d 56
	3	CB 5F	CB 58	CB 59	CB 5A	CB 5B	CB 5C	CB 5D	CB 5E	DD CB d 5E	FD CB d 5E
	4	CB 67	CB 60	CB 61	CB 62	CB 63	CB 64	CB 65	CB 66	DD CB d 66	FD CB d 66
	5	CB 6F	CB 68	CB 69	CB 6A	CB 6B	CB 6C	CB 6D	CB 6E	DD CB d 6E	FD CB d 6E
	6	CB 77	CB 70	CB 71	CB 72	CB 73	CB 74	CB 75	CB 76	DD CB d 76	FD CB d 76
	7	CB 7F	CB 78	CB 79	CB 7A	CB 7B	CB 7C	CB 7D	CB 7E	DD CB d 7E	FD CB d 7E
RESET BIT 'RES'	0	CB 87	CB 80	CB 81	CB 82	CB 83	CB 84	CB 85	CB 86	DD CB d 86	FD CB d 86
	1	CB 8F	CB 88	CB 89	CB 8A	CB 8B	CB 8C	CB 8D	CB 8E	DD CB d 8E	FD CB d 8E
	2	CB 97	CB 90	CB 91	CB 92	CB 93	CB 94	CB 95	CB 96	DD CB d 96	FD CB d 96
	3	CB 9F	CB 98	CB 99	CB 9A	CB 9B	CB 9C	CB 9D	CB 9E	DD CB d 9E	FD CB d 9E
	4	CB A7	CB A0	CB A1	CB A2	CB A3	CB A4	CB A5	CB A6	DD CB d A6	FD CB d A6
	5	CB AF	CB A8	CB A9	CB AA	CB AB	CB AC	CB AD	CB AE	DD CB d AE	FD CB d AE
	6	CB B7	CB B0	CB B1	CB B2	CB B3	CB B4	CB B5	CB B6	DD CB d B6	FD CB d B6
	7	CB BF	CB B8	CB B9	CB BA	CB BB	CB BC	CB BD	CB BE	DD CB d BE	FD CB d BE
SET BIT 'SET'	0	CB C7	CB C0	CB C1	CB C2	CB C3	CB C4	CB C5	CB C6	DD CB d C6	FD CB d C6
	1	CB CF	CB C8	CB C9	CB CA	CB CB	CB CC	CB CD	CB CE	DD CB d CE	FD CB d CE
	2	CB D7	CB D0	CB D1	CB D2	CB D3	CB D4	CB D5	CB D6	DD CB d D6	FD CB d D6
	3	CB DF	CB D8	CB D9	CB DA	CB DB	CB DC	CB DD	CB DE	DD CB d DE	FD CB d DE
	4	CB E7	CB E0	CB E1	CB E2	CB E3	CB E4	CB E5	CB E6	DD CB d E6	FD CB d E6
	5	CB EF	CB E8	CB E9	CB EA	CB EB	CB EC	CB ED	CB EE	DD CB d EE	FD CB d EE
	6	CB F7	CB F0	CB F1	CB F2	CB F3	CB F4	CB F5	CB F6	DD CB d F6	FD CB d F6
	7	CB FF	CB F8	CB F9	CB FA	CB FB	CB FC	CB FD	CB FE	DD CB d FE	FD CB d FE

Fig. 2.11 Grupo de manipulación de bits

			CONDITION									
			UN- COND.	CARRY	NON CARRY	ZERO	NON ZERO	PARITY EVEN	PARITY ODD	SIGN NEG	SIGN POS	REG B≠0
JUMP 'JP'	IMMED. EXT.	nn	C3 n n	DA n n	D2 n n	CA n n	C2 n n	EA n n	E2 n n	FA n n	F2 n n	
JUMP 'JR'	RELATIVE	PC+e	18 e-2	38 e-2	30 e-2	28 e-2	20 e-2					
JUMP 'JP'	REG. INDIR.	(HL)	E9									
JUMP 'JP'		(IX)	DD E9									
JUMP 'JP'		(IY)	FD E9									
'CALL'	IMMED. EXT.	nn	CD n n	DC n n	D4 n n	CC n n	C4 n n	EC n n	E4 n n	FC n n	F4 n n	
DECREMENT B, JUMP IF NON ZERO 'DJNZ'	RELATIVE	PC+e										10 e-2
RETURN 'RET'	REGISTER INDIR.	(SP) (SP+1)	C9	D8	D0	C8	C0	E8	E0	F8	F0	
RETURN FROM INT 'RETI'	REG. INDIR.	(SP) (SP+1)	ED 4D									
RETURN FROM NON MASKABLE INT 'RETN'	REG. INDIR.	(SP) (SP+1)	ED 45									

NOTE–CERTAIN FLAGS HAVE MORE THAN ONE PURPOSE. REFER TO SECTION 6.0 FOR DETAILS

Fig. 2.12 Instrucciones de salto y de bifurcación

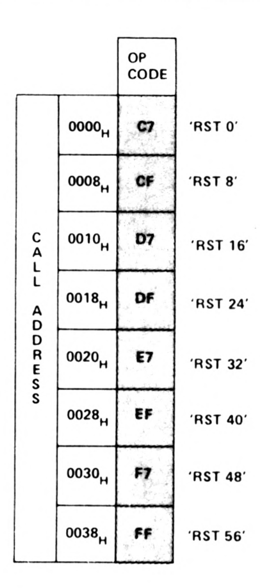


Fig. 2.13 Instrucciones de restart

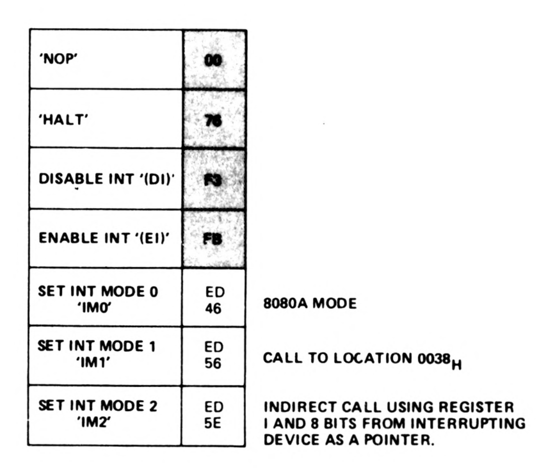


Fig. 2.14 Instrucciones de control del Z80-CPU

			SOURCE PORT ADDRESS	
			IMMED.	REG. INDIR.
			(n)	(C)
INPUT 'IN'	REG ADDRESSING	A	DB n	ED 78
		B		ED 40
		C		ED 48
		D		ED 50
		E		ED 58
		H		ED 60
		L		ED 68
'INI' – INPUT & Inc HL, Dec B	REG. INDIR	(HL)		ED A2
'INIR'– INP, Inc HL, Dec B, REPEAT IF B≠0				ED B2
'IND'– INPUT & Dec HL, Dec B				ED AA
'INDR'– INPUT, Dec HL, Dec B, REPEAT IF B≠0				ED BA

BLOCK INPUT COMMANDS (the last four rows)

Fig. 2.15 Instrucciones de entrada

			SOURCE							
			REGISTER							REG. IND.
			A	B	C	D	E	H	L	(HL)
'OUT'	IMMED.	(n)	D3 n							
	REG. IND.	(C)	ED 79	ED 41	ED 49	ED 51	ED 59	ED 61	ED 69	
'OUTI' – OUTPUT Inc HL, Dec b	REG. IND.	(C)								ED A3
'OTIR' – OUTPUT, Inc HL, Dec B, REPEAT IF B≠0	REG. IND.	(C)								ED B3
'OUTD' – OUTPUT Dec HL & B	REG. IND.	(C)								ED AB
'OTDR' – OUTPUT, Dec HL & B, REPEAT IF B≠0	REG. IND.	(C)								ED BB

(REG. IND. / (C) columns: PORT DESTINATION ADDRESS; OUTI, OTIR, OUTD, OTDR rows: BLOCK OUTPUT COMMANDS)

Fig. 2.16 Instrucciones de salida

3. Conexión de componentes del sistema

3.1 Decodificación:

Tal como se ha mencionado en el capítulo sobre el hardware del Z80, un microprocesador solo no puede todavía utilizarse para aplicaciones prácticas. Más bien deberá conectarse con otros componentes del sistema, como pueden ser la memoria y componentes periféricos. En principio ésto es muy sencillo, ya que únicamente deberán unirse las conexiones del bus de datos y del bus de control con las conexiones correspondientes del bus del microprocesador del Z80, haciendo que el componente se convierta en conversacional mediante una conexión apropiada de decodificación del Z80 con una dirección determinada, o bien dentro de un campo de direcciones concreto. El microprocesador Z80 posee 16 salidas físicas de direcciones, de la A0 hasta la A15, con las que podrán formarse 2^{16} = 65536 direcciones distintas. En este campo se depositan normalmente los componentes de la memoria del sistema, mientras que los componentes periféricos se encuentran generalmente en la página cero (0 hasta 255) del campo de direcciones I/O (I/O = Input/Output = entrada/salida). El campo de direcciones I/O es únicamente conversacional con las instrucciones correspondientes I/O, de modo que la página cero del campo de direcciones de la memoria podrá ser ocupada independientemente del campo de direcciones I/O. Para el direccionamiento de los componentes periféricos se requieren por lo tanto únicamente 8 líneas de dirección, utilizando las direcciones A0 hasta A7.

Los componentes periféricos raramente ocupan más de cuatro direcciones I/O; sin embargo, los componentes de la memoria requieren tantas direcciones como posiciones de memoria posean. Si por ejemplo deseamos poner un

componente en el campo 0000H hasta 07FFH (inclusive), procederemos de la siguiente manera:

El campo de 0000H hasta 07FFH corresponde a $2048 = 2^{11}$ direcciones de memoria que pueden seleccionarse individualmente. Tal como lo demuestra la siguiente representación, pueden generarse 2^{11} direcciones con 11 bits.

	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0
0000H :	0	0	0	0	0	0	0	0	0	0	0
0001H :	0	0	0	0	0	0	0	0	0	0	1
.											
.											
.											
.											
07FFH :	1	1	1	1	1	1	1	1	1	1	1

El Z80 no trabaja con 11 bits de dirección sino con 16, de modo que mediante una conexión adecuada (conexión de decodificación) deberá asegurarse que el componente sea únicamente contactado cuando los bits de dirección restantes (A11 hasta A15) sean igual a cero. Si éste no fuera el caso, entonces la posición de memoria cero del componente no sería únicamente contactada con la dirección de memoria 0000H, sino también con las direcciones 0800H, 1000H, 1800H, 2000H, etc.:

	A15	A14	A13	A12	A11	A10	A0
0000H :	0	0	0	0	0	0	0
07FFH :	0	0	0	0	0	1	1
0800H :	0	0	0	0	1	0	0
08FFH :	0	0	0	0	1	1	1

Para la solución de este problema se tienen a disposición componentes de decodificación integrados, como por ejemplo el 74LS138.

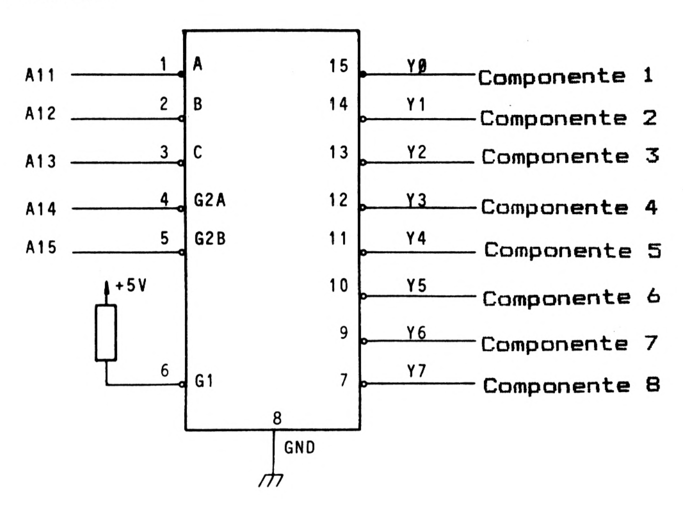


Fig. 3.1 Decodificador 74LS138

Las entradas G1, G2A y G2B sirven para la elección del mismo 74LS138. Mediante ellas es fácilmente posible la conexión en cascada de hasta 8 de estos componentes de decodificación. Con ello podrían definirse y posicionarse hasta 8 x 8 = 64 componentes en el área de la memoria. Las entradas A, B y C son las entradas propias de decodificación. Con estas tres entradas pueden formarse 2^3 = 8 distintas direcciones. Estas direcciones activan una de las ocho salidas del componente de

decodificación, con ello la salida elegida se pone en cero lógico (los pequeños círculos del componente significan que el componente de esta posición trabaja con lógica cero activada). Las salidas están unidas con las correspondientes entradas de selección GS o CE de los componentes, y las activan únicamente cuando aparece una dirección válida para ellas en el bus de direcciones.

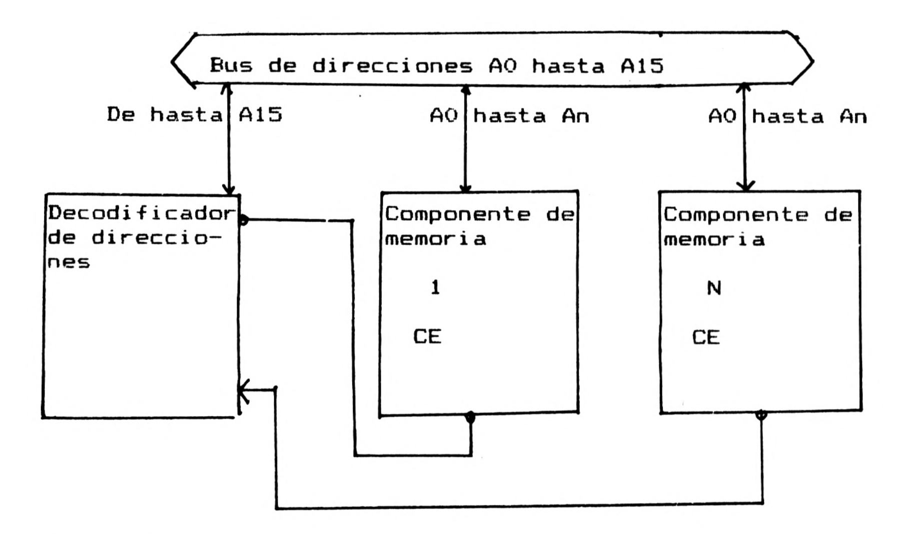


Fig. 3.2 Principio de la decodificación completa para componentes de memoria.

Si nos quedamos en nuestro ejemplo y deseamos poner el componente en otra área, por ejemplo de 2800H hasta 2FFFH, únicamente tendremos que conectarlo al pin 10 en

lugar del pin 15. Esto resulta de la tabla de la verdad de la decodificación, si se piensa que en el área 2800H hasta 2FFFH los estados de A11, A12 y A13 son lógicos 1,0 y 1.

A B C	Componente elegido	Pin	Salida
0 0 0	1	15	Y0
0 0 1	2	14	Y1
0 1 0	3	13	Y2
0 1 1	4	12	Y3
1 0 0	5	11	Y4
1 0 1	6	10	Y5
1 1 0	7	9	Y6
1 1 1	8	7	Y7

Fig. 3.3 Tabla de la verdada para el componente de decodificación 74LS138

El método tratado hasta ahora sobre la selección de los componentes utiliza la decodificación completa. En sistemas muy pequeños puede utilizarse un método más sencillo, llamado "decodificación lineal". En lugar de decodificar líneas de dirección se utilizan las mismas líneas de dirección para la selección de los componentes, con lo cual con cada componente a ser elegido se reducirá a la mitad el campo de memoria disponible de 2^{16} = 65536. La ventaja está en la eliminación del componente de decodificación que normalmente se exije.

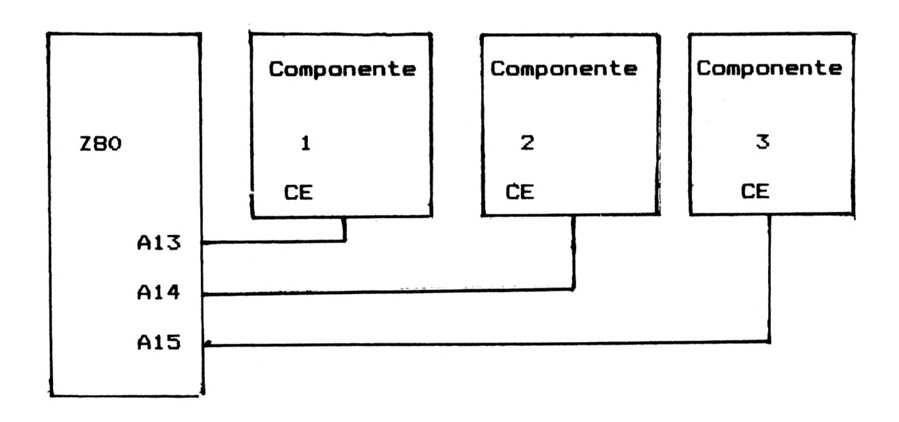


Fig. 3.4 Principio de la decodificación lineal (para 3 componentes)

En este ejemplo se seleccionan los tres componentes con las líneas de dirección A15, A14 y A13. El campo de memoria restante (A0 hasta A12) es de únicamente 2^{13} = 16K direcciones de memoria.

Antes de fijar las direcciones de memoria para los distintos componentes, se realiza normalmente un plan de ocupación de memoria (Memory Map). Esto tiene la ventaja de que a simple vista puede apreciarse en qué campo de dirección se encuentra la parte determinada. Incluso se detectan de inmediato espacios vacíos e interferencias.

Dirección	Contenido
FFFH – 1800H	Libre
17FFH – 1000H	2K RAM 2
0FFFH – 0800H	2K RAM 1
07FFH – 0000H	2K ROM

Fig. 3.5 Ejemplo para el plan de ocupación de memoria (Memory Map).

Para la representación del plan de ocupación I/O bastará una recopilación en forma de listado de los componentes I/O con sus correspondientes direcciones I/O.

Después de haber mencionado el principio de la decodificación, podremos conectar algunos componentes típicos al microprocesador Z80.

3.2 Conexión de componentes de memoria:

a) Conexión de ROM's

Las ROM's son componentes de memoria que mantienen su contenido una vez realizada la programación. Por este motivo se les llama también "memoria de valores fijos". Aquí no se pretende detallar los distintos tipos de memoria, pero sí dar la oportunidad al lector de tener una visión general sobre las propiedades y campos de aplicación de los tipos de memoria más importantes. Las ROM's (Read Or Memory) se aplican generalmente para almacenar programas, mientras que la memoria de trabajo que sirve de interfase de datos leídos o resultados de proceso, consta de RAM's (controles).

Las ROM's se dividen fundamentalmente en las siguientes categorías:

ROM: El contenido de la memoria está programado en forma de máscara, o sea, que el fabricante lo indica concretamente. Fabricar tales ROM's expresamente sólo vale la pena si se trata de fabricar un gran número de ellas. Ejemplos de programación para ROM's son los programas monitor, los generadores de caracteres, las tablas de función, las tablas de linealización, etc.

PROM: El contenido de la memoria puede ser programado por el usuario. La programación se realiza generalmente mediante la cauterización de tramos de diodos. El proceso es irreversible. Las PROM's pueden utilizarse cuando las ROM's no son rentables y el programa ya no tiene que modificarse.

EPROM: El contenido de la memoria puede ser programado por el usuario, borrado mediante luz UV y programarse eléctricamente de nuevo. Los contenidos se mantienen durante muchos años y exteriormente pueden reconocerse de inmediato por su ventana de cuarzo. Las EPROM's se utilizan a menudo cuando se trata de cantidades pequeñas a medianas. Incluso se encuentran ya en el mercado un gran número de programadores EPROM de precio favorable, de modo que la aplicación de estas memorias permanentes también puede ser interesante para el que programa como pasatiempo.

EEPROM/ EAROM: El contenido de la memoria puede asímismo ser eléctricamente programado por el usuario. Contrariamente a la EPROM, estas memorias pueden ser borradas eléctricamente. Ya que estas memorias son más caras que las EPROM, se aplican generalmente para "salvar" datos importantes de la RAM (por ejemplo cuando hay algún fallo en la red). Una rutina determinada carga el contenido de la memoria RAM (o partes de la misma) en la EEPROM correspondiente al bajar la tensión de alimentación por debajo de un valor critico determinado.

La distribución de conexiones de los distintos tipos de ROM es bastante compatible entre sí - con pocas excepciones -, de modo que normalmente puede ponerse una EPROM en el zócalo de una ROM programada en forma de máscara y viceversa (si la capacidad de memoria es la misma). Las ROM's trabajan con una extensión de palabra de 8 bits. Se leen cuando el decodificador de direcciones

las elige y al mismo tiempo las señales de control del Z80 MREQ y RD están activas.

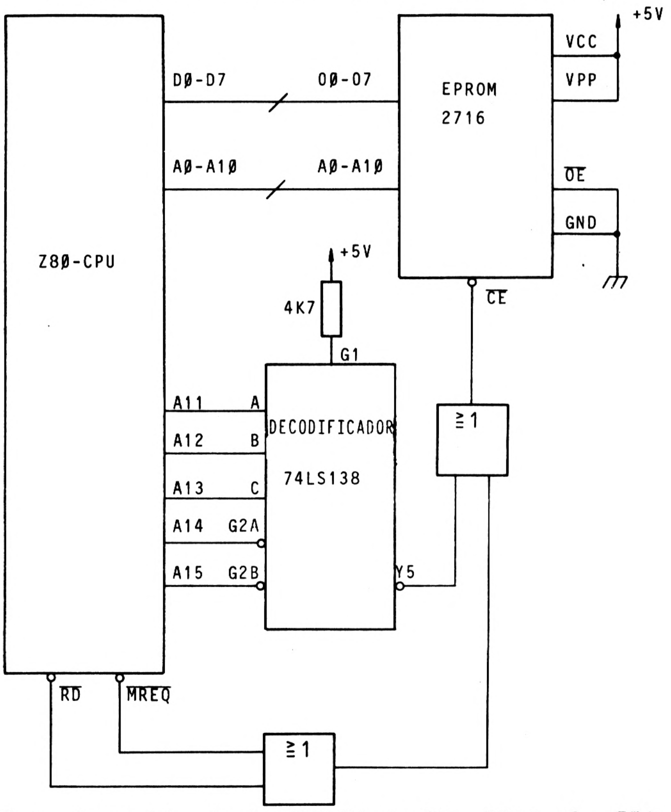


Fig. 3.6 Conexión de la EPROM de 2K 2716 al Z80 en el área de memoria 2800H a 2FFFH.

Para la conexión de varias ROM's se ponen las distintas líneas de selección de la memoria del componente decodificador en las conexiones CE correspondientes de las ROM's.

Especialmente, en sistemas mayores, las salidas del Z80 serían sobrecargadas, de modo que la aplicación de buffers de dirección (por ejemplo 74LS244) sería necesaria. Los buffers de dirección "refuerzan" el bus de direcciones procedente del Z80. Además, en sistemas mayores, las ROM's no pueden impulsar directamente el bus de datos sino que deberán ser apoyadas por buffers de datos de la memoria (74LS244).

b) Conexión de RAM's

Las RAM's (Random Access Memory) son componentes de memoria que conservan únicamente su contenido mientras se les suministra la tensión correspondiente. Hay que distinguir entre RAM's estáticas y dinámicas. Mientras que las RAM's estáticas pueden grabarse y leerse sin ningún tipo de medida adicional, las dinámicas tienen que ser "refrescadas" a lo más tardar cada 2 ms para que no pierdan su contenido. Normalmente, para este refresco se exige una instalación especial de hardware, pero el microprocesador Z80 tiene ya una lógica de refresco incorporada que emite en los momentos adecuados las RAM's necesarias en la mitad inferior del bus de direcciones, junto con el dispositivo de control RFSH. Contrariamente a las ROM's, las RAM's pueden mostrar una amplitud de palabra distinta. Normalmente se utilizan RAM's con una amplitud de 1 bit, 4 y 8 bits, aunque las RAM's dinámicas prefieren la amplitud de palabra de 1 bit. La amplitud exigida de 8 bits para el Z80 se obtiene mediante la conexión en paralelo de varias RAM's.

RAM's estáticas:

Ya que las RAM's no pueden únicamente ser leídas (como las ROM's), sino que también pueden ser grabadas, para poder seleccionar los componentes RAM se requiere un circuíto ampliado:

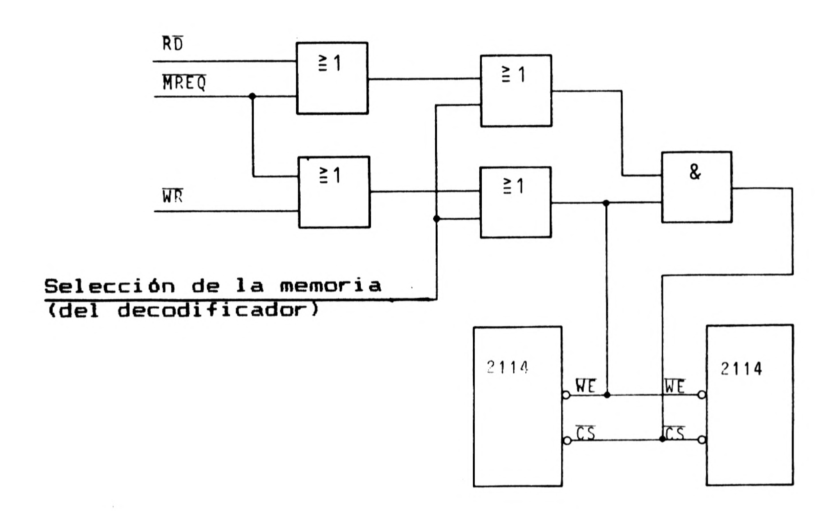


Fig. 3.7 Modo de conexión de las entradas WE y CS de dos RAM's 2114.

Las RAM's tienen únicamente una sola línea, con la que se gobierna la dirección de la transferencia de datos (datos del Z80 en la RAM = grabar, y datos de la RAM al Z80 = leer / inglés: grabar = write y leer = read). El estado lógico de esta línea determina la dirección de transferencia. Si con RAM's estáticas se requiere un buffer del bus de datos, se podría aplicar aquí, por ejemplo, el buffer bidireccional del bus de datos 74LS245.

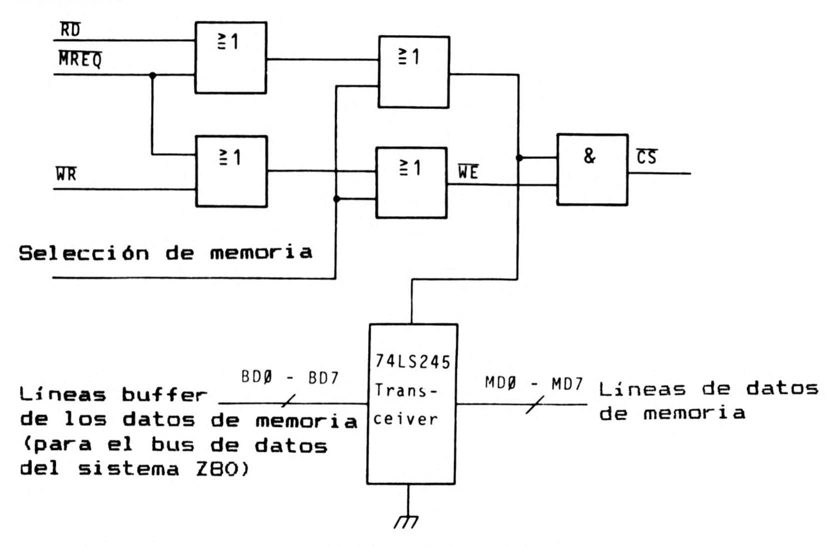
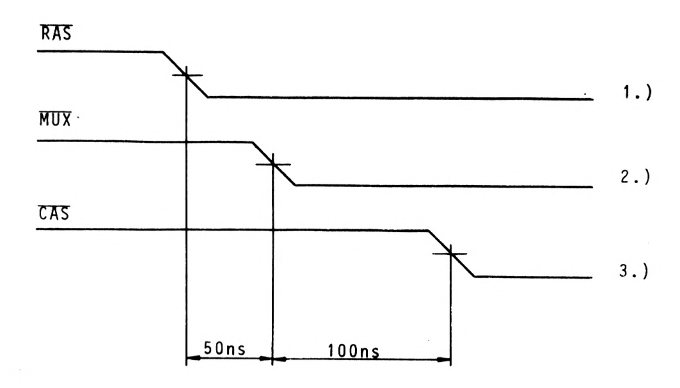


Fig. 3.8 Tamponamiento del bus de datos

RAM's dinámicas:

Se utilizan raramente en ordenadores de control; sin embargo, las llamadas RAM's dinámicas se encuentran cada vez más en los ordenadores personales. Son más económicas que las RAM's estáticas, especialmente a partir de capacidades de 16 bytes y reúnen en poco espacio una enorme capacidad de memoria. Usuales son por ejemplo memorias de 16K-bits, 64K-bits, 256K-bits y 1M de bits. Memorias con 4M de bits podrán encontrarse en el mercado en un próximo futuro. Para poder mantener de modo reducido el número de pins de conexión y el tamaño constructivo de la memoria, se multiplexan las direcciones de memoria dinámicas, es decir, que cada dirección se pone en dos partes al mismo tiempo y una tras otra en el componente. La parte de direcciones que se lee en este momento es determinada por el estado de la señal multiplex (MUX). Si MUX es lógico 1, la parte inferior de la dirección estará en la memoria y si es MUX lógico 0, entonces estará la parte superior. Otras dos señales, la CAS (Columm Adress Strobe) y RAS (Row Adress Strobe), indican la validez de las correspondientes partes de las direcciones (CAS = 0 para la mitad superior de las direcciones, RAS = 0 para la mitad inferior).



1.) Parte inferior de la dirección válida
2.) Cambiar a la parte superior de la dirección
3.) Parte superior de la dirección válida

Fig. 3.9 Proceso temporal de las señales RAS, CAS y MUX

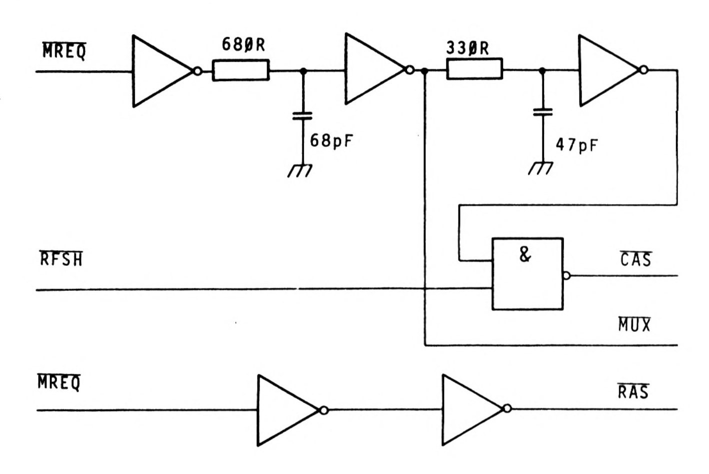


Fig. 3.10 Ejemplo para la generación de RAS, CAS y MUX

Si se aplican memorias dinámicas en un sistema Z80 deberá tenerse en cuenta que el Z80 suministra las líneas de direcciones de refresco necesarias sólo mientras ejecuta instrucciones. En el caso de un reset o con DMA, las memorias no se refrescarán más, perdiendo así su contenido si un circuito adicional no asegura los ciclos de refresco.

Conexión de componentes periféricos (componentes de entrada/salida)

Los componentes periféricos posibilitan al sitema Z80 tener contacto con el "mundo exterior". Ejemplos para tales componentes son el Z80-PIO y el Z80-SIO. Los componentes periféricos del Z80 son contactados mediante instrucciones especiales de entrada/salida. Siempre que se ejecuta una instrucción E/S, el Z80 activa la línea IORQ. Junto con la línea RD respectivamente WR puede desencadenarse un proceso de lectura o de escritura de hasta un componente de E/S. Las operaciones de memoria no se activan con el IORQ sino que lo hacen con el MREQ. Para los componentes de entrada/salida no están disponibles las 16 líneas de direccionamiento sino que se tendrán únicamente las 8 inferiores A0 hasta A7 para realizar la decodificación. Esto tiene sin embargo la desventaja de una limitación a máximo 256 ports I/O, pero aporta la ventaja de disponer de un proceso más rápido de instrucciones mediante formatos más cortos de las mismas. Prácticamente, casi nunca se requieren más de 256 Ports. Aquí deberá mencionarse que un componente de E/S (componente periférico) puede ocupar varias direcciones (por ejemplo para el canal 1 leer, canal 1 grabar. canal 2 leer, etc.).

La selección de componentes se efectúa según los mismos principios que se han descrito al inicio del capítulo.

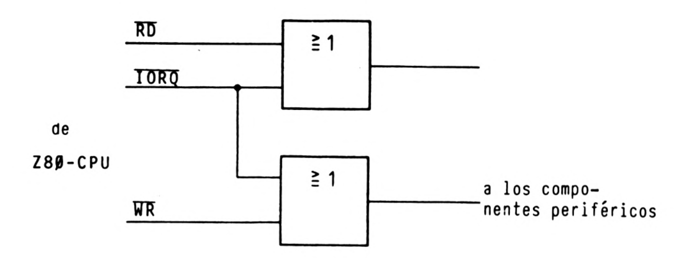


Fig. 3.11 Creación de las señales IOR e IOW

Las señales definitivas de port, lectura o grabación, se consiguen mediante el enlace OR de la señal de decodificación y de las señales IOR respectivamente IOW.

A base de un ejemplo se representará la decodificación de un componente típico periférico. Partiremos de la base que el componente ocupa cuatro direcciones de E/S (8CH-8FH). Como decodificador podrá servir un 74LS138. El circuito correspondiente se muestra en la Fig. 3.12.

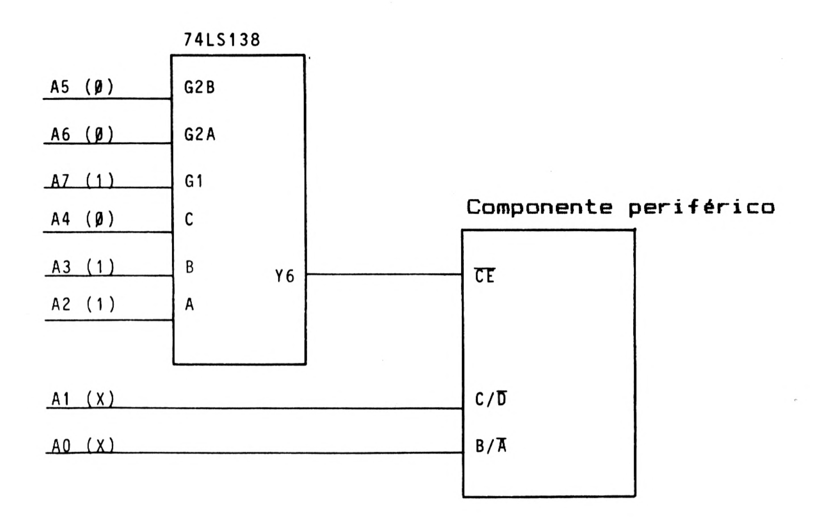


Fig. 3.12 Decodificación de un componente periférico típico.

4. Técnicas de entrada/salida

4.1 Instrucciones de entrada/salida:

Las instrucciones de entrada del Z80-µP son:

IN A,(N): La N es la dirección de 8 bits (A0 hasta A7) del port a ser leído.

IN A,(C): El registro C contiene la dirección de ocho bits (A0 hasta A7) del port a ser leído.

IND: Esta es una instrucción automática de transferencia de un byte. El port seleccionado indirectamente a través del registro C se cargará en el elemento indirectamente direccionado de la memoria a través del par de registros HL y a continuación se disminuirá el par de registros HL y el registro B. La B sirve de contador. Después de ejecutar la instrucción estará todo preparado para la nueva lectura del port y almacenaje en forma de tabla del byte leído en la memoria.Esta instrucción es, por ejemplo, adecuada para bucles de lectura port en los que después de cada proceso de lectura se realizan todavía otras operaciones (bucles de espera, verificación derazonabilidad, etc.).

INI: Para esta instrucción equivale lo mencionado para IND con la excepción de que el par de registros HL no disminuye sino que es aumentado. La posición de memoria más baja direccionada de la tabla confeccionada mediante esta instrucción que se ha ejecutado reiteradas veces, contiene la inscripción más

antigua de la tabla (con la instrucción IND la más nueva).

INDR: Esta es una instrucción automática de transferencia de componentes. Corresponde en

su función a la instrucción IND, con la diferencia que dicha instrucción se repite automáticamente tantas veces hasta que el contenido del registro B se ponga a cero. La instrucción INDR es especialmente adecuada cuando tiene que leerse un port hasta 256 veces consecutivas y los datos tengan que almacenarse en una tabla de la memoria. Si el Z80-µP trabaja con una frecuencia de 4 MHz, el port a ser leído deberá estar en condiciones de preparar los datos en un espacio de tiempo de máximo 5,25 µs.

INIR: Para esta instrucción sirve lo indicado en INDR con la excepción que el par de registros HL no disminuye sino que aumenta.

Las instrucciones de salida del Z80-µP son:

OUT A,(N): La N es la dirección de ocho bits (A0 hasta A7) del port, al que debe emitirse el contenido del acumulador A.

OUT A,(C): El registro C contiene la dirección de ocho bits (A0 hasta A7) del port, al que debe emitirse el contenido del acumulador A.

OUT D: Esta es una instrucción automática de transferencia de un yte.

El par de registros HL direcciona indirectamente aquella posición de memoria a la que debe emitirse el port direccionado a través del registro C. El registro B sirve de contador y finalmente disminuye como el par de registros HL. Repitiendo la ejecución de esta instrucción podrá emitirse una tabla secuencialmente al port. La instrucción se presta, por ejemplo, para bucles de salida en los que después de cada instrucción de salida se ejecutan otras operaciones (por ejemplo bucles de espera, transposiciones de códigos, etc.).

OUT I: Para esta instrucción sirve lo indicado para la OUT D con la excepción de que el par de registros HL no disminuye, sino que aumenta.

OTDR: Esta es una instrucción automática de transferencia de componentes. Corresponde en su función a la instrucción OUT D, con la diferencia que esta instrucción se ejecuta automáticamente hasta que el contenido del registro B se ponga a cero. La instrucción OTDR se presta especialmente para cuando deben emitirse hasta 256 bytes de una tabla de la memoria a un port (por ejemplo: inicialización de componentes). Con una frecuencia de 4 MHz en el Z80-µP, el port deberá estar en condiciones de asumir un nuevo byte cada 4 µs.

OTIR: Para esta instrucción sirve lo indicado para OTDR, con la excepción de que el par de registros HL no disminuye, sino que aumenta.

Todas las instrucciones de entrada/salida están construidas de cara a la velocidad. Utilizan direcciones de ocho bits y les basta el formato de instrucciones de 2 Bytes. El primer byte contiene el código de operaciones y el segundo o bien la dirección Port de 8 Bits (IN A,(N), OUT (N),A) o la segunda mitad del código de operación. Debido a las direcciones port de únicamente ocho bits de ancho, se ha fijado un límite del número de ports direccionados a un máximo de 256; sin embargo, en la práctica es suficiente.

Handshaking

Cuando se trata de una comunicación asincrónica con dispositivos de entrada/salida deberá asegurarse que la comunicación tenga lugar cuando ambos comunicandos (ordenador y elemento de E/S) estén ya dispuestos. Esto puede realizarse ya sea mediante la técnica del "Handshaking" ("apretón de manos") o bien a través de la técnica del interrupt ("técnica de interrupción").

El proceso del handshaking, al efectuar la entrada, es el siguiente:

a) Z80 al componente de E/S: ¿Tienes un carácter válido para mí?

b) Componente de E/S al Z80: Sí/No

El Z80 podrá leer un carácter válido del componente de E/S únicamente cuando éste haya dado el "sí" correspondiente. El componente de E/S posee para el handshaking

un registro especial de estado. El Z80-µP lee (normalmente) el bit 7 de este registro de estado, recibiendo información sobre si está en condiciones de leer el carácter o no. Si no, se repetirá la consulta hasta conseguir el éxito deseado.

Ejemplo: Aquí debe leerse un dato de un elemento de entrada a través de un componente de entrada.

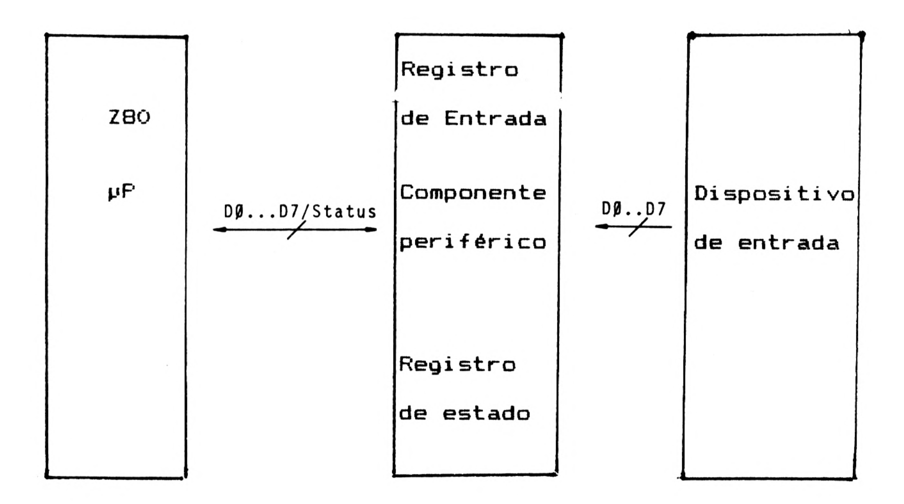


Fig. 4.1 Entrada de datos con Handshake

El programa correspondiente podría ser como sigue:

```
ESPERA:    IN(STATUS); lee registro de estado
           BIT 7,A   ; ¿está el carácter válido dispues-
                       to?
           JR Z,ESPERA; si no, repite consulta
           LD A(DATUM);  carga el registro de entrada en
                         el accu
```

El dato se encuentra ahora en el acumulador y puede ser así debidamente procesado. Posteriormente, con el adecuado salto a ESPERA se repetirá el proceso de lectura.

El proceso de handshaking al efectuar la salida es:

a) Z80 al componente E/S: ¿Estás dispuesto para leer?
b) Componente E/S al Z80: Sí/No

El Z80-µP puede emitir un dato (carácter) al componente E/S, cuando éste haya cancelado la consulta. El Z80-µP consulta al bit 7 del registro de estado del componente E/S hasta que el componente E/S esté dispuesto para aceptar los datos.

Ejemplo: Aquí deberá emitirse un dato a un dispositivo de salida a través de un componente de salida.

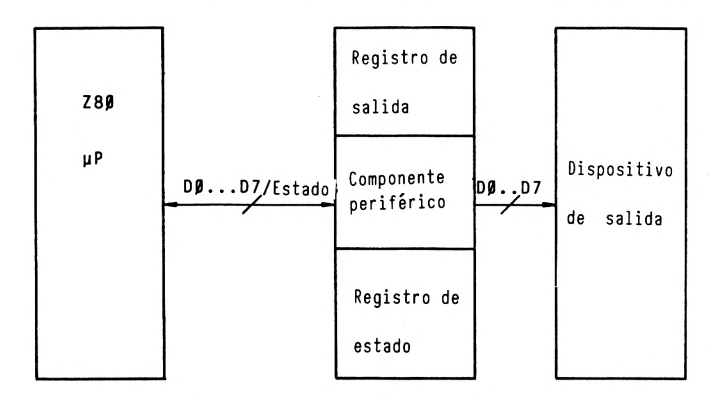


Fig. 4.2 Emisión de datos con handshake.

El programa correspondiente podría ser según sigue:

```
ESPERA: IN A,(STATUS)      ; lee registro de estado
        BIT 7,A            ; ¿preparado para la recepción?
        JR Z,ESPERA        ; si no, repite la consulta
        LD A,(FECHA)       ; carga fecha en el accu
        OUT (SALIDA),A     ; emitir fecha
```

La fecha se encuentra ahora en el registro de salida del componente periférico y seguirá aquí hasta que quede anulada por una nueva fecha. Dado el caso y después de recorrer

un bucle de retardo, podrá emitirse la próxima fecha (carácter) mediante un salto a ESPERA.

4.2 Polling:

Si se debe atender a más de un dispositivo de E/S por el procesador, entonces podrán tenerse tres técnicas básicas presentes para el tratamiento de este problema: polling, interrupt y DMA. Polling (consulta) es el más sencillo de los tres métodos y puede aplicarse para operaciones que no estén sujetas a un tiempo determinado. Cada componente periférico posee generalmente un registro de estado. Cuando un dispositivo periférico solicita una función a través del procesador, el bit 7 del registro de estado correspondiente la activará. El procesador pasa por un bucle de consultas y verifica cada vez el bit 7 del registro de estado correspondiente. Si está activado el bit 7 de registro de estado, el procesador saltará a la rutina de servicio asociada a este dispositivo.

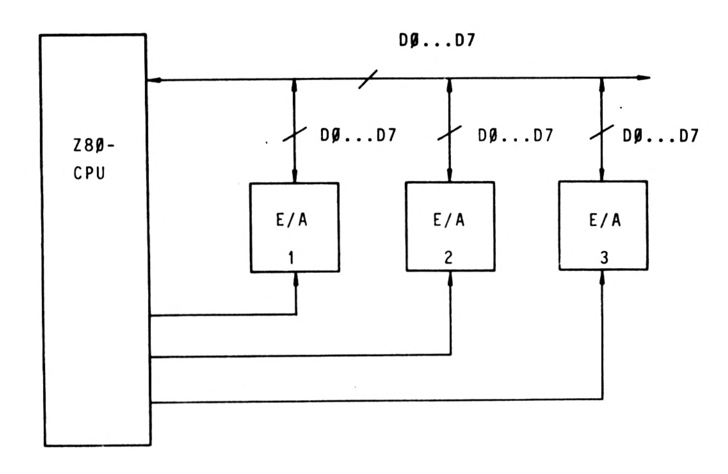


Fig. 4.3 Control de entradas/salidas de 3 dispositivos mediante polling.

El programa correspondiente podría ser como sigue:

```
POLLING:IN A,(STATUS 1) ;  lee  registro  de  estado del
        BIT 7 ,A          dispositivo 1
        CALL NZ, DIS1   ; ¿Petición de servicio?
        IN A,(STATUS 2) ; Salto a la rutina de servicio
        BIT 7 ,A        ; Dispositivo 2
        CALL NZ, DIS2
        IN A,(STATUS 3) ; Dispositivo 3
        BIT 7,A
        CALL NZ, DIS3
        JR POLLING      ; Repetir la consulta
```

El polling no requiere ninguna ayuda adicional del hardware y es, además, en cuanto al software, muy sencillo de realizar, de modo que el polling se aplica cuando la utilización de los dispositivos periféricos no es crítica y todas las tareas del procesador se limitan al servicio de los dispositivos periféricos conectados.

4.3 Interrupts:

Los interrupts (interrupciones) tienen la ventaja frente a los polling, que el procesador no tiene que estar consultando contínuamente a los dispositivos, sino mientras realiza otros trabajos, hasta que un dispositivo comunica al procesador mediante un interrupt que desea ser servido.

El Z80 posee tres entradas distintas de interrupt: INT, NMI y BUSRQ. El BUSRQ se menciona en el apartado "DMA".

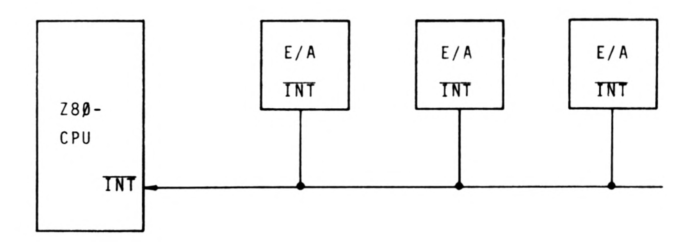


Fig. 4.4 Controles de entrada/salida con interrupts.

La mayoría de componentes de E/S poseen salidas de Interrupt que mediante circuitos de salida open-collector o bien open-drain pueden ser directamente conectadas a la llamada línea de interrupt.

Esta línea de interrupt se comunica con la entrada INT del Z80-µP. Cuando por lo menos se activa una salida de Interrupt, entonces se dispara un Interrupt (una interrupción) del programa en curso. Por último deberá constatarse qué dispositivos de E/S de los que están conectados ha disparado el interrupt. Incluso para ello existen varias posibilidades. Por ejemplo, puede aplicarse el polling y después de cada interrupt consultar secuencialmente los dispositivos E/S, fijándose la prioridad de los dispositivos mediante la consecución temporal de la consulta. Otro método aplica componentes especiales de control de prioridad. Estos fijan consecutivamente en el bus de datos, después de haberse realizado el interrupt, la dirección completa de 16 Bits de la rutina de servicio correspondiente al mismo, desde donde el Z80-µP la lee. Una solución intermedia es la llamada técnica del "Daisy-Chain" (Daisy-Chain = cadena de margaritas). Los componentes específicos del sistema Z80, como pueden ser PIO, SIO, CTC, etc., están dispuestos especialmente para este tipo tan efectivo de comprobación del dispositivo demandante.

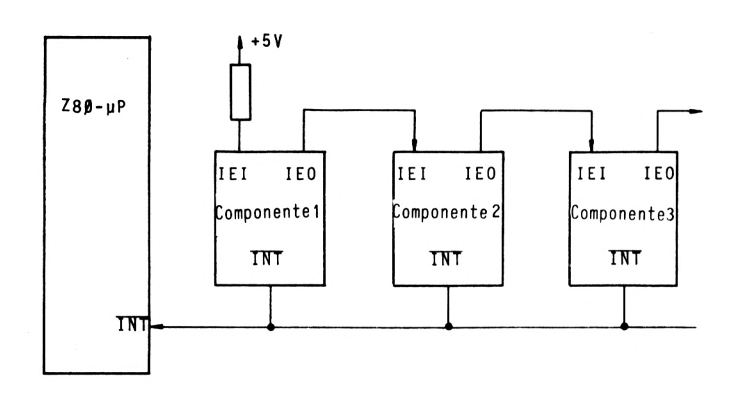


Fig. 4.5 Principio de la cadena de prioridad Daisy-Chain en el sistema Z80.

Tan pronto como un componente liberado de E/S activa su salida de interrupt y ocasiona en el Z80-µP un interrupt, el procesador finaliza el programa después de haber concluído la instrucción en curso, salva el contador del programa PC en la pila y confirma el interrupt recibido mediante la activación lógica a cero del IORQ y M1. Todos los componentes de E/S que puedan desencadenar interrupts se proveen con señales de control IORQ y M1, pudiendo así reconocer la confirmación del interrupt del Z80 (INTA- Interrupt-Acknowledge). Aquel componente de E/S que haya sido liberado para los interrupts depositará un vector de direccionamiento

de ocho bits en el bus de datos. La liberación funciona de la siguiente forma:

Cada componente puede ocasionar un interrupt. Sin embargo, tan pronto un componente haya ocasionado un interrupt, todos los componentes siguientes de la cadena quedarán desactivados para nuevos interrupts. Los interrupts que hayan sido desactivados no se "pierden", sino que quedan almacenados hasta ser liberados. La restitución se realiza tan pronto el componente de mayor prioridad haya reconocido la instrucción especial de retroceso RETI del Z80-µP. Entonces desactiva su demanda de interrupt y libera los componentes de menor prioridad.

Un sistema Z80 con utilización de la técnica Daisy-Chain desarrolla, junto con el modo de interrupt 2 del Z80-µP, su completa capacidad. Mientras el componente interruptor periférico suministra la mitad inferior del vector de direccionamineto, la mitad superior se encuentra en el registro de interrupt del Z80. Con la dirección resultante podrá direccionarse a cualquier posición del campo de memoria de 64K una tabla de saltos, que, a su vez, indicará las distintas rutinas de servicio de interrupt (véase también la descripción de instrucciones IM2 en el capítulo "Instrucciones completas del Z80"). La entrada INT del Z80 puede ser enmascarada con la instrucción DI y liberada nuevamente mediante la instrucción EI. La entrada interrupt NMI del Z80 - contrariamente al INT - no es, en cuanto al software, enmascarable. Se utiliza generalmente para salvar datos importantes en una memoria residente (por ejemplo EEPROM) en el caso de que la tensión de alimentación quede por debajo de un valor crítico, ocasionando así un NMI el salto a la dirección 0066H. Aquí se encuentra la rutina correspondiente de salvación. Contrariamente a la entrada INT, que trabaja de modo estático, la entrada NMI funciona dinámicamente; ésta se

disparará al caer el flanco de la señal de entrada NMI. Durante el ciclo NMI se almacenará el último estado INT, cargando el interrupt flip-flop IFF1 en el IFF2. Con el fin de poder enmascarar interrupts en la entrada INT durante el ciclo NMI, se fijará el IFF1 a cero. Después de la ejecución del NMI se volverá a crear el estado inicial del INT cargando del IFF2 al IFF1. Al final de la rutina NMI deberá constar la instrucción RETN para que busque la dirección PC y se vuelva a crear el estado lógico del IFF1 flip-flop.

Al principio de una rutina de servicio de interrupt deberán salvarse todos los registros del Z80 que serán modificados por la rutina de servicio. Una rutina típica de servicio para un interrupt enmascarable está construida de la siguiente manera:

```
PUSH AF
PUSH BC         Salvar el registro universal
PUSH DE         en la pila
PUSH HL


                Verdadera rutina de servicio del
                interrupt

POP  HL
POP  DE         Buscar en la pila el registro
                universal
POP  BC
POP  AF
EI              Liberación de los interrupts
                después del salto de retorno.
RETI            Salto de retorno a la rutina
                de servicio
```

Si el programa de servicio utiliza también los pares de registros IX e IY, éstos deberán asimismo ser salvados.

Si se trata de un interrupt (NMI) no enmascarable, la rutina de servicio correspondiente, con excepción de la instrucción del salto de retorno RETI, es idéntica a la de arriba. En lugar de RETI se utiliza la instrucción especial de retorno RETN que además carga IFF2 después de IFF1.

Los componentes especiales del sistema Z80 reconocen la ejecución de una instrucción RETI desactivando seguidamente la demanda de interrupt. La instrucción del salto de retorno RET no es reconocida por estos componentes, quedando consecuentemente al final de estas rutinas que no son las de interrupt. Si se utilizan componentes periféricos de otros fabricantes, deberá tenerse en cuenta que su demanda de interrupt deberá ser desactivada mediante la indicación de una palabra de control correspondiente porque no valoran la instrucción especial RETI del Z80.

Otra posibilidad de poder salvar contenidos importantes del registro en caso de un interrupt está en el intercambio de ambos juegos de registros universales del Z80 por la secuencia de instrucciones:

```
EX   AF,AF'
EXX
```

Interrupts anidados que aparecen cuando una rutina de servicio de interrupt activada es interrumpida mediante un interrupt de mayor prioridad, no pueden ser manipulados cambiando el juego universal registros del Z80.

En este caso que prácticamente puede aparecer cuando al principio de la rutina de servicio de interrupt se liberan otros interrupts (por lo menos dos dispositivos deberán ser capaces de interrumpirse), deberá aplicarse el método PUSH-POP. Es también importante reservar tanto espacio de memoria para la pila como grado superior de anidamiento del interrupt se requiera para salvar todos los contenidos del registro.

Ejemplo: La rutina de servicio 1 utiliza los registros AF,BC,HL e IX.
La rutina de servicio 2 utiliza los registros AF,BC,DE,HL.
La rutina de servicio 3 utiliza los registros AF,HL.

El grado máximo de anidamiento del interrupt es de 3. La suma de todos los registros de 8 bits a ser salvados es, en este ejemplo, de 20. Además deberán tenerse en cuenta 2 x 3 = 6 bytes para salvar los tres contenidos del contador del programa; la suma será de 26 bytes de espacio de memoria requeridos para la pila. (El ejemplo parte de que el programa principal no contiene subprogramas.)

Ahora deberán representarse específicamente las propiedades de los dos interrupts INT y NMI:

a) NMI:

Este interrupt no es enmascarable, o sea que será siempre aceptado.

- El NMI es disparado por flanco, o sea que para su inicialización basta un pequeño impulso.

- Si un NMI es activado, ocurrirá lo siguiente:

1.) El contenido del PC se salvará en la pila.
2.) El estado del interrupt flip-flop IFF1 se cargará después del IFF2 (salvará).
3.) El IFF1 se pondrá a cero, cerrando así otros Interrupts (en la entrada INT).
4.) El PC se cargará con 0066H. (El programa saltará por lo tanto a la dirección 0066H, donde se encuentra la rutina de servicio NMI.)

- La rutina de sevicio NMI finaliza con la instrucción especial de retorno RETN que se encargará de recuperar el antiguo estado del PC y del IFF1.

b) INT:

- Esta entrada de interrupt podrá ser mascarada (componente ada) de acuerdo con el software mediante la instruc ción DI (IFF1 = 0). La liberación se realiza con la instrucción EI (IFF1 = 1).

- La entrada INT es sensitiva al estado en que se encuentra; deberá ser activada hasta ser aceptada por el Z80.

- La entrada INT podrá trabajar en 3 modalidades. Las instrucciones para estas 3 modalidades son IM0, IM1 e IM2. Después de cada reset del Z80 (o sea también después de cada conexión) se seleccionará automáticamente la modalidad 0. La modalidad 0 corresponde a la modalidad de interrupt del 8080 µP.

1.) Modalidad de interrupt 0 (IM0)

Después de haberse disparado un interrupt, el procesador confirmará la aceptación del mismo, activando al mismo tiempo las líneas de control IORQ y M1. Seguidamente leerá

el Bus de datos. El componente a ser interrumpido deberá poner en el bus de datos ya sea una instrucción RST o un CALL. Si el procesador lee una instrucción RST, saltará a la dirección de la correspondiente instrucción RST (existen ocho instrucciones distintas RST). El campo de direccionamiento RST va desde 0000H hasta 0038H en aumentos de 0008H respectivamente.

Si el componente a ser interrumpido manda una instrucción CALL, el Z80-µP leerá otros dos bytes del bus de datos que deberán ponerse a disposición por el componente periférico y serán interpretados como dirección de salto para la rutina de servicio.

2.) Modo interrupt 1 (IM1)

En este modo de interrupt el Z80-µP bifurca después de un interrupt a la dirección 0038H, donde empieza la correspondiente rutina de servicio.

3.) Modo de interrupt 2 (IM2)

Aquí se encuentra la fuerza real del tratamiento del interrupt del Z80-µP. En primer lugar deberá cargarse el registro I del Z80-µP con la mitad superior del vector de direccionamiento que indicará hacia la tabla de saltos de la memoria. En el caso de efectuarse un interrupt, el componente periférico a ser interrumpido deberá poner la mitad inferior del vector en el bus de datos. El procesador bifurca a continuación a aquella dirección que se encuentra en la tabla de saltos, direccionando de esta manera la tabla de saltos mediante el vector (o mejor la dirección en la tabla de saltos).

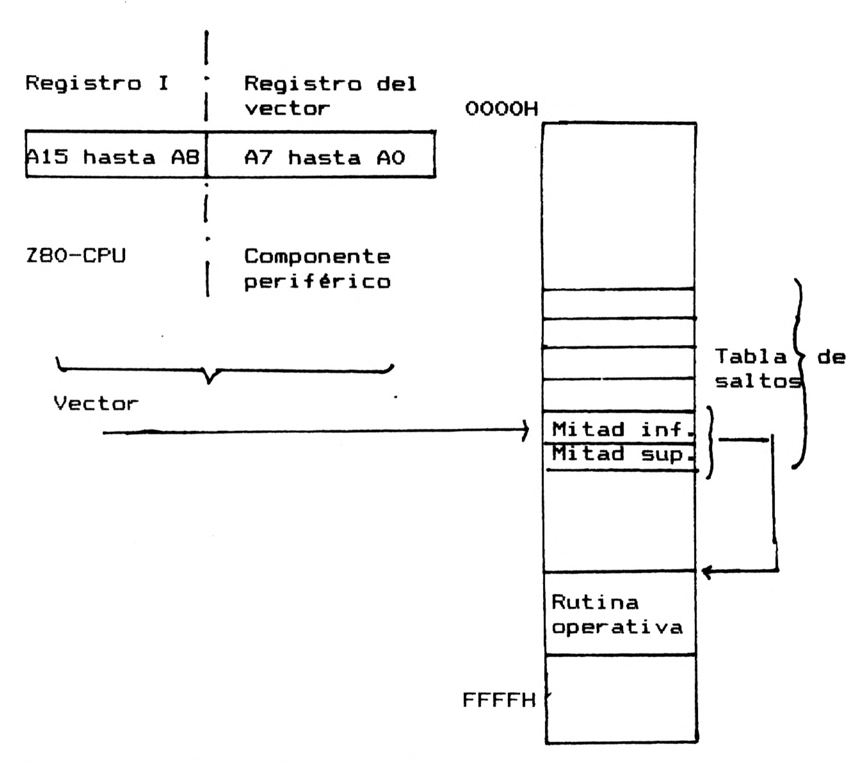


Fig. 4.6 Interrupts del vector

Tanto el lugar de la tabla de saltos como también el lugar de las rutinas operativas de la memoria es libremente elegible mediante una programación adecuada. del registro I y de los registros del vector.

4.4 DMA (Direct Memory Access)

Los interrupts posibilitan una rápida reacción sobre las exigencias operativas de los componentes periféricos. El propio servicio se realiza mediante rutinas de Software. Algunos dispositivos periféricos, como por ejemplo los Floppy-Disks, exigen una transferencia de datos mucho más rápida que lo que se podría realizar con software. En estos casos se aplica la técnica del acceso directo a la memoria (DMA). Para ello se dispone de componentes especiales, los llamados DMAC'S (Direct Memory Access Controller). Estos DMAC'S contienen muchas veces varios canales, asignándoles a cada uno de ellos un dispositivo periférico. Los dispositivos periféricos no dirigen sus solicitudes de interrupt al Z80-µP, sino al DMAC. El DMAC interrumpe el procesador a través de la entrada BUSRQ, aceptando el impulsor de direcciones y del bus de datos del procesador un estado de alto ohmiaje (flouting). Tan pronto como el Z80-µP haya aceptado el BUSRQ (final de la transferencia actual del bus), éste lo confirmará a través del BUSAK. El DMAC sabrá entonces que los buses de direccionamiento y de datos son de alto ohmiaje e inicia su tarea accediendo directamente a la memoria del sistema. Las transferencias de datos realizadas por el DMAC están dirigidas a base del hardware y por ello son muy rápidas. Cuando el DMAC anula el BUSRQ, el Z80-µP recupera el control sobre el bus de direccionamiento y de datos.

El DMA se aplica, cuando deben transferirse datos rápidamente que, o bien se encuentran en la memoria del sistema o deberán ser cargados en la misma.

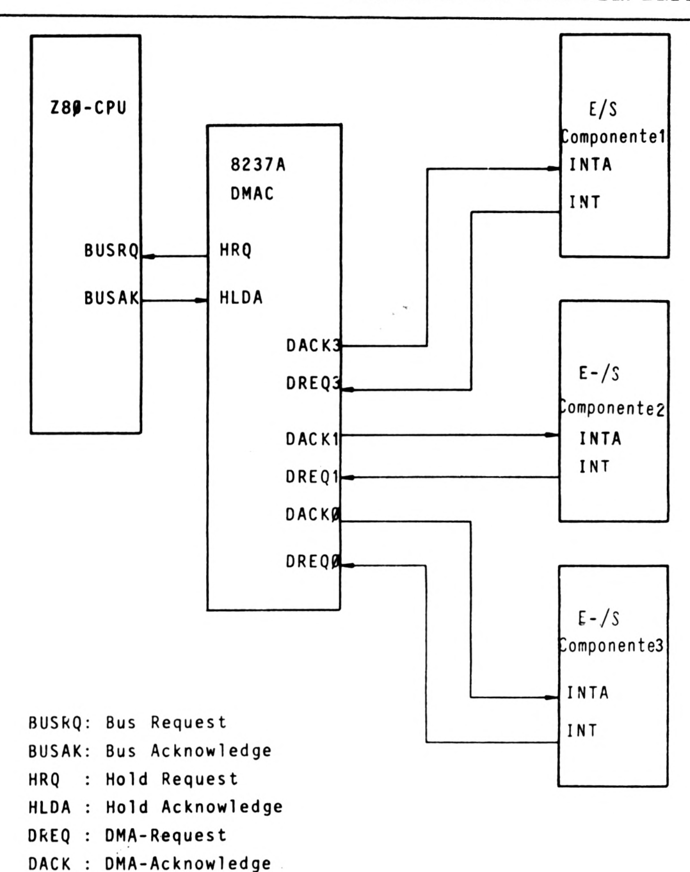


Fig. 4.7 Microprocesador Z80 con el 8237 DMAC (Intel)

Además, existen procesadores especiales de entrada/salida. Estos tienen menos utilidad para un intercambio rápido de datos, que para una preparación acelerada de datos asignados a un dispositivo periférico. Los procesadores de entrada/salida (I/O - Processors) aligeran el trabajo del procesador. Están equipados con una instrucción de entrada/salida especialmente efectiva y se programan con microprocesadores.

Existe un gran número de componentes especiales que están preparados para una función determinada, como por ejemplo el Floppy-Disk Controller, los coprocesadores aritméticos, los coprocesadores de texto, el Controller CTR, los componentes GPIB, etc. La descripción de estos componentes especiales sobrepasaría el marco de este libro en demasía; además, su representación no se exige para la comprensión de las técnicas de entrada/salida, ya que las mismas han sido debidamente presentadas.

5. Transmisión de datos en paralelo

5.1 Generalidades:

El Z80-µP alimenta un bus de datos de ocho bits (D0 hasta D7); un procesador lee bajo el correspondiente proceso de lectura la información que se encuentra en el D0 hasta D7, escribiendo asimismo, según el proceso de grabación, los datos que se hallan en el D0 hasta el D7. Así pues, se transmitirán en paralelo ocho bits de datos. Contrariamente a ello trabaja la transmisión serial de datos con una anchura de bit de 1, es decir, transmitiendo únicamente bit a bit en una línea de datos.

La transmisión de datos en paralelo permite lógicamente una velocidad de transmisión más elevada que la serial; sin embargo, también requiere más líneas para la transmisión. Para distancias cortas la necesidad de un mayor número de líneas no tiene gran importancia; sin embargo, si se trata de grandes distancias, por motivos de coste, sólo se considera la transmisión serial. La comunicación entre el procesador y los componentes del sistema conectados (por ejemplo la memoria, PIO, SIO...) se realiza a través del bus de datos común en paralelo. Unidades periféricas externas como la impresora, el plotter, etc., pueden trabajar tanto con transmisión de datos en paralelo como en serie.

Los fabricantes de unidades periféricas (impresoras, etc.) tienen un interés vital en la compatibilidad de sus dispositivos con muchos sistemas de ordenadores (distintos), de modo que se han desarrollado "estándares para las posiciones de corte".

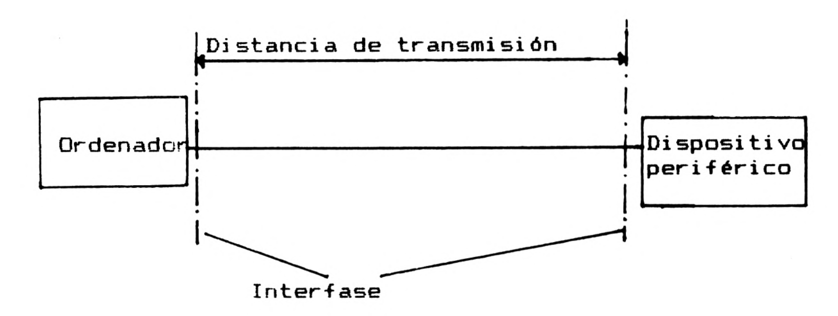


Fig. 5.1 Interfases

Si ambas interfases coinciden, podrán transferirse datos entre el ordenador y el dispositivo periférico. En el campo de la transmisión de datos en paralelo la interfase-Centronic y el bus IEC (IEEE-488, GPIB) son los más importantes a ser mencionados. La interfase Centronic se utiliza frecuentemente en impresoras y se caracteriza por su sencilla manipulación. La transmisión se realiza generalmente bajo el código ASCII de 7 Bits, eventualmente, incluso con bit de paridad. El bus IEC encuentra ante todo aplicación en el campo industrial y sirve para la conexión de hasta 15 dispositivos como son el voltímetro, fuentes de alimentacion, Wobbler, etc., a una distancia de hasta 20 m. con una velocidad de transmisión de máximo 1Mbit/s. La interfase IEC trabaja con una técnica complicada de handshake, teniendo a disposición los bloques adecuados para el apoyo de este estándar (por ejemplo los bloques GPIB de la firma Intel o bien bloques GPIA de la firma Motorola). Una descripción exacta del estándar se encuentra en el IEEE 488-1978.

5.2 <u>Centronics:</u>

Esta interfase puede ser activada mediante las señales de mando STROBE (traspaso) y ACKNOWLEDGE (confirmación). El ordenador activa las líneas con los datos correspondientes y además la línea Strobe, indicando de esta manera al periférico (p.e. impresora) que éste acepte los datos. El siguiente dato será transmitido, en el momento que el dispositivo periférico haya indicado la recepción del dato al activar la línea ACKNOWLEDGE. Un programa adecuado seguirá consultando la línea ACKNOW-

LEDGE, hasta que ésta se active (cero lógico) antes de mandar el carácter. Al utilizar un Z80-PIO puede fácilmente realizarse una interfase con un sólo port (por ejemplo port A). Mientras que A0 hasta A6 (resp. A0 hasta A7 con bit de paridad) transmiten los caracteres ASCII, ARDY y ASTB servirán para el handshake (el ARDY indica que un carácter válido está disponible y a través de ASTB el dispositivo periférico confirma la aceptación de un dato). Sin embargo, port A deberá estar programado en la modalidad 0 (sólo salida).

5.3 IEEE 488 (GPIB, HPIB, IEC 62):

Este estándar se desarrolló a principios de los años 70 por la Firma Hewlett Packard y corresponde en su versión de 1978 a su último estado. Mientras que hasta entonces todo usuario industrial debería "arreglárselas" para componerse sus propias interfases para los distintos instrumentos, actualmente puede contarse con un amplio abanico de aparatos compatibles con el IEEE 488. A pesar de que H.P. tenga sus derechos de patente sobre este método, los derechos de licencia a ser compensados son limitados en comparación con los gastos de desarrollo que se exigen. El estándar es tan universal que con el mismo pueden prácticamente cubrirse todas las exigencias industriales (a distancias de hasta 20m). Con el mismo se definen todas las condiciones de interfases mecánicas, eléctricas y funcionales, de modo que cada aparato de cualquier fabricante puede ser aplicado siempre y cuando disponga de una interfase IEC. Los planos de

cables, definiciones de clavijas de contacto, etc. son superfluos por su normativa. Incluso aquellos aparatos que no requieren mando alguno a través de un ordenador, pueden comunicarse entre si a través del bus de IEC. Tanto los interrupts como los DWA están previstos por el estándar. La comunicación se realiza asincrónicamente, de modo que incluso los aparatos que tienen cuotas de transmisión muy diferentes pueden ser conectados. Los datos se transmiten byte a byte (también los procesadores de 16 y 32 bits se comunican con los dispositivos periféricos byte a byte).

El estándar distingue tres funciones básicas, pudiendo un aparato realizar cada una de las mismas (o incluso combinaciones):

1. Controller: El controller dirige otras unidades (Listener y Talker).

2. Talker: Un talker da información al controller.

3. Listener: Un listener recoge información de un controller.

Un ordenador une, por ejemplo, las tres funciones de talk, listen y control, mientras que un generador de señales únicamente contiene la función de listen. Los distintos aparatos están paralelamente conectados al bus de IEC. Lo interesante es que el estándar no posee ningún bus de direccionamiento. Las direcciones (8 bits) son asímismo transmitidas como los datos (8 bits) y las instrucciones (7 bits) en el bus de datos (8 bits). Las transmisiones en el bus de datos serán dirigidas mediante las tres líneas de mando NECRD, DAV y NDAC (servicio de acuse de recibos, handshake de 3 líneas).

Otras cinco líneas de mando (ATN,EQI,SRQ,IFC y REN) definen el estado general del sistema.

El bus de IEC consta por lo tanto de:

8 líneas de datos
3 líneas de mando de handshake y
5 líneas generales de mando.

Las tres líneas de mando de handshake son:

NRFD: Not Ready for Data. Un listener indica que todavía no está preparado para recibir datos o instrucciones. Todos los listener deberán retirar su mensaje de NRFD antes de que una nueva transmisión de datos pueda tener lugar en el bus.

NDAC: Not Data Accepted. Un listener indica que todavía no ha recibido el byte de datos, respectivamente de instrucciones. Para el controller no se habrá terminado la transmi sión mientras un listener indique NDAC.

DAV: Data Valid. Un talker indica que tiene datos a disposición válidos.

Un ciclo de acuse de recibo tiene el siguiente aspecto:

1.) No existe ningún dato válido en el bus de datos (DAV = lógico 1). Ningún listener está preparado (NRFD = lógico 0). Ningún listener ha aceptado datos (NDAC = lógico 0).

2.) Un dato es puesto en el bus de datos.
3.) Todos los listener están preparados (NRFD =lógico 1).
4.) La validez del dato será indicada (DAV = lógico 0).
5.) Los listener desactivan los NRFD (NRFD = lógico 0).
6.) Todos los listener han aceptado los datos (NDAC = lógico 1).
7.) El talker desactiva el DAV (DAV = lógico 1).
8.) Los listener desactivan el NDAC (NDAC = lógico 0).

Con ello se habrá conseguido nuevamente el estado inicial del punto 1 y el próximo dato podrá ser transmitido.

Formulado de modo más sencillo, el intercambio de acuse de recibo funciona de la siguiente manera: Con NRFD se indicará el inicio de la transmisión y con NDAC el final de la misma. Los puntos de tiempo exactos serán determinados mediante los bordes de impulso descendentes, respectivamente ascendentes, del DAV. Condición para una transmisión es la puesta a punto de un dato correspondiente sobre las líneas de datos DIO 1 hasta DIO 8.

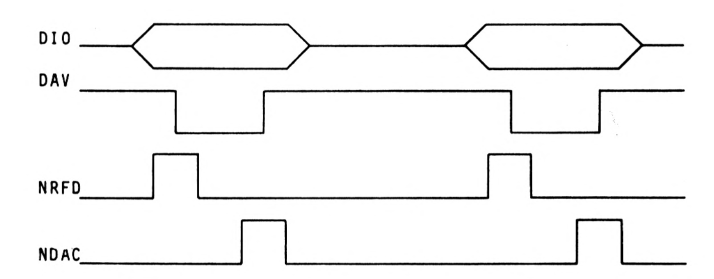


Fig. 5.2 Handshake de 3 líneas IEC

Las cinco líneas generales de control son:

ATN: Attention. El controller activa estas líneas para indicar que ha puesto en el bus de datos un byte de direccionamiento o bien de instrucciones.

EOI: End Or Identify. Un talker indica el final de una transmisión de datos. Un controller inicia, junto con el ATN, un polling en paralelo (en el caso de un interrupt).

SRQ: Service Request. Esta es la línea de interrupción del sistema. Cada unidad podrá comunicar al controller que desea ser abastecida.

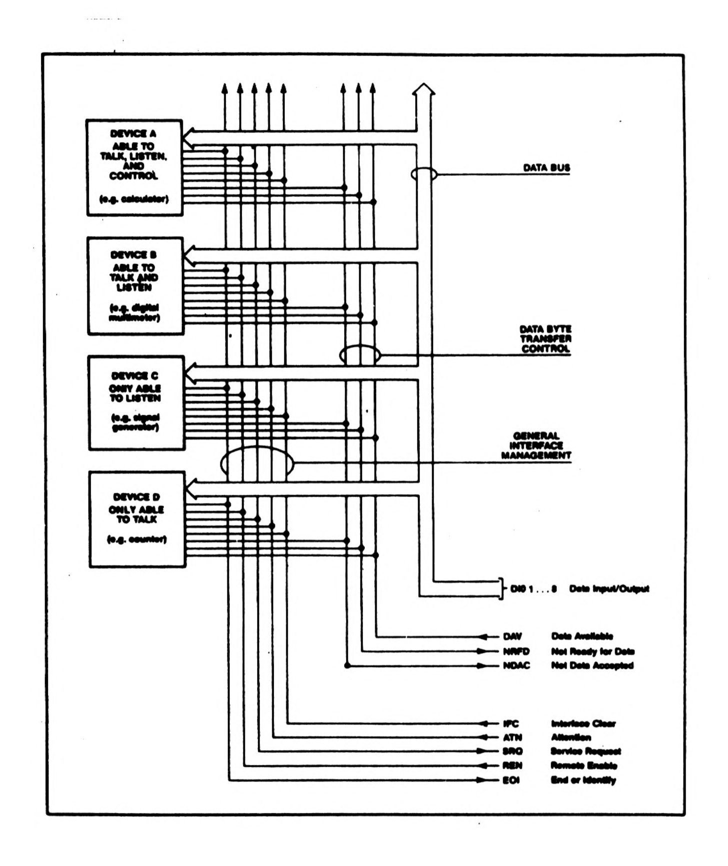
IFC: Interface Clear. El controller del sistema podrá aquí inicializar todas las demás unidades con un estado definido de inicio.

REN: Remote Enable. El controller posibilita aquí que la unidad direccionada trabaje mediante control remoto (remote), es decir, atender únicamente al us de IEC y no a cualquier elemento de servicio de placas frontales.

Eléctricamente, el estándar trabaja con niveles de TTL. Como conectores por enchufe se aplican enchufes industriales de 24 polos, tales como Ampherol Series 57 Micro-Ribbon.
La conexión es la siguiente:

Nº del PIN:
Función:

1	DIO1
.	.
.	.
4	DIO4
5	EOI
6	DAV
7	NRFD
8	NDAC
9	IFC
10	SRQ
11	ATN
12	PANTALLA
13	DIO5
.	.
.	.
16	DIO8
17	REN
18	Masa
.	.
.	.
24	Masa


DEVICE A
ABLE TO
TALK, LISTEN,
AND
CONTROL
(e.g. calculator)
DEVICE B
ABLE TO
TALK AND
LISTEN
(e.g. digital
multimeter)
DEVICE C
ONLY ABLE
TO LISTEN
(e.g. signal
generator)
DEVICE D
ONLY ABLE
TO TALK
(e.g. counter)
DATA BUS
DATA BYTE
TRANSFER
CONTROL
GENERAL
INTERFACE
MANAGEMENT
DIO 1 . . . 8 Data Input/Output
DAV Data Available
NRFD Not Ready for Data
NDAC Not Data Accepted
IFC Interface Clear
ATN Attention
SRQ Service Request
REN Remote Enable
EOI End or Identify

Además de las tres funciones fundamentales de talk, list y control existen además otras siete funciones de IEC (Source Handshake, Acceptor Handshake, Service Request, Remote Local, Parallel Poll, Device Clear y Device Trigger).

En cuanto a información más detallada, nos remitimos a la norma IEEE-488-1978.

5.4 Z80-PIO (Input/Output en paralelo):

El Z80-µP no puede comunicar directamente con dispositivos periféricos. Para ello necesita los llamados componentes de comunicación. El Z80-PIO representa uno de estos componentes para la transmisión de datos en paralelo. Está ideado para la aplicación en los sistemas Z80 (2,5 MHz). Para los sistemas Z80A (4MHz) se dispone del Z80A-PIO; sus propiedades se asemejan funcionalmente al Z80-PIO. El Z80-PIO (PIO = Input/Output paralelo) tiene dos ports de entrada/salida; a cada port se le han asignado dos líneas de handshake y la función de los mismos es programable. La mayoría de los dispositivos periféricos corrientes, como son la impresora, el lector de cintas perforadas, aparatos de programación PROM, etc., pueden ser conectados al Z80-PIO sin tener que hacer conexiones adicionales. A pesar de que puedan aplicarse también otros componentes en paralelo de entrada/salida (por ejemplo 8255 de INTEL), la utilización del Z80-PIO es por lo general más ventajosa, al menos por favorecer por completo las posibilidades de interrupt del Z80-µP (interrupts del vector). Toda transferencia de datos entre la CPU y el dispositivo periférico se efectúa bajo handshakes dirigidos por el interrupt. Además, pueden encadenarse entre si hasta cuatro PIO's en una "Daisy Chain" sin tener que hacer conexiones com-plementarias, consiguiéndose así una

prioridad simple de interrupt. En la modalidad 3 del PIO puede desencadenarse finalmente un interrupt si en las entradas se encuentra (previamente programada) una información determinada (especialmente de interés para aplicaciones de control de proceso).

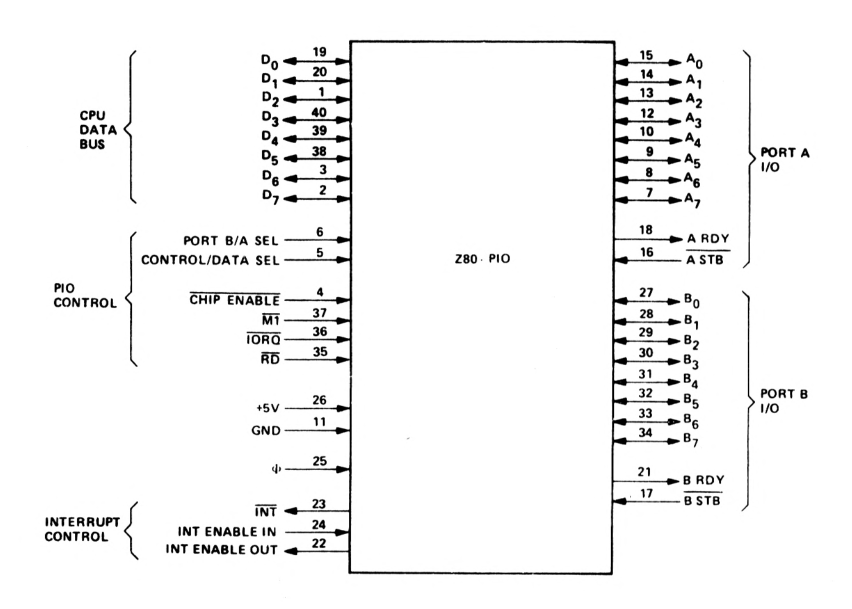


Fig. 5.4 Disposición de los pins del Z80-PIO

Descripción de los pins del Z80-/Z80A-PIO:

D7 hasta D0	bus de datos del Z80-µP, bidireccional, tristate. En estas líneas se intercambian los bytes de datos y de instrucciones entre el CPU y el PIO.
B/A SEL	(Port B/A Select) A través de esta entrada se elige uno de ambos ports, B o bien A (port B - lógico 1, port A - lógico 0). Esta entrada se conecta generalmente con la A0 del bus de direcciones CPU.
C/D SEL	(Control/Data - Select) Si esta entrada es lógico 1, se activa con el registro de código de control del port seleccionado a través de B/A SEL; si la entrada es lógico 0, los registros de datos del port correspondiente serán activados. La entrada define, pues, el tipo de transferencia entre la CPU y el PIO: transferencia de instrucciones o de datos. Generalmente se conecta con la A1 del bus de direcciones CPU.
CE	(Chip Enable) Si esta entrada es lógico 0, el PIO podrá comunicarse con la CPU. La entrada se maneja generalmente a través de un circuito de decodificación (desde A7 hasta A2). Un Z80-p10 ocupa cuatro direcciones port (port A, port B / transferencia de instrucciones, transfe rencia de datos).
ϕ	(Systemtakt) Aquí se conecta el reloj del sistema (máx. 2,5 MHz para el Z80-PIO y máx. 4 MHz para el Z80A-PIO)

M1 (Machine Cicle 1) Esta entrada se conecta con el M1 del Z80-μP y realiza cuatro funciones:

1. Si M1 y RD son lógico 1, el PIO reconocerá un ciclo de instrucciones de la CPU.
2. Si M1 e IORQ son lógico 0, el PIO reconoce la confirmación del interrupt de la CPU (importante en la instrucción RETI).
3. M1 sincroniza la lógica del interrupt.
4. El PIO se desactiva cuando M1 es lógico 0, pero los RD e IORQ son lógico 1. (El Z80-PIO no posee ninguna entrada especial de RESET.)

IORQ (Input/Output Request) Esta entrada se conecta con la IORQ del Z80-μP. Una transferencia entre la CPU y el PIO tiene únicamente lugar cuando IORQ y CE son lógico 0. Cuando IORQ y M1 son al mismo tiempo lógico 0 (CPU confirma la exigencia del interrupt), el PIO pondrá el vector del interrupt de 8 bits del port que interrumpe en el bus de datos de la CPU, siempre y cuando este port posea la máxima prioridad de interrupt.

RD (Read) Esta entrada se conecta con el RD del Z80-μP. La CPU podrá leer el PIO cuando el IORQ, el RD y el CE sean lógico cero. (El registro PIO que será leído, quedará concretado por la situación del B/A SELECT y del C/D SELECT.)

IEI (Interrupt Enable In) Esta entrada se requiere cuando más de un componente puede ocasionar interrupts dentro de una cadena de

prioridad Daisy-Chain. Unicamente cuando esta entrada es lógico 1, el PIO podrá crear interrupts (entonces se habrá "liberado" para los interrupts).

IEO (Interrupt Enable Out) Esta salida se utiliza como la IEIO, en relación con la lógica de prioridad de la Daisy-chain. Se conecta con la IEI del siguiente eslabón y bloquea así todos los componentes subsiguientes (con prioridad inferior), mientras sea lógico 0, el IEO será únicamente lógico 1, cuando tanto el IEI del PIO sea lógico 1 como PIO no sea atendido por la CPU con una rutina de servicio del interrupt.

INT (Interrupt Request) Esta salida del Open-Drain será conectada con la entrada INT del Z80-µP, así como con las salidas INT de los demás componentes.

A0 hasta A7 Este bus de 8 bits sirve para la comunicación entre el port A del PIO y el dispositivo periférico. Pueden intercambiarse informaciones de mando, de estado y de datos (dependiendo del tipo de servicio (=modalidad)).

ASTB (Port A Strobe) Impulso Strobe (= recepción) del dispositivo periférico al Port A. La función de esta entrada, dependiendo de la modalidad programada, es la siguiente:

1.) Modalidad 0: El dispositivo periférico confirma mediante el borde de impulso ascendente la recepción de de los datos PIO.
2.) Modalidad 1: El dispositivo periférico exige al PIO, leer el dato que está preparado.
3.) Modalidad 2: El dispositivo periférico exige al PIO, emitir el contenido del registro de salida del port A al A0 hasta el A7. El borde de impulso ascendente del ASTB indica al PIO que el dispositivo periférico ha aceptado el dato.
4.) Modalidad 3: La entrada ASTB se bloquea internamente.

ASTB es siempre low estando activo.

ARDY (Register A Ready) Esta salida envía una señal de disponible al dispositivo periférico, donde la función de la salida depende de la modalidad programada (=modalidad funcional):

Modalidad 0: Cuando ARDY es lógico 1, el dispositivo periférico podrá leer el registro de salida del port A (a través de A0 hasta A7).

Modalidad 1: Cuando ARDY es lógico 1, el dispositivo periférico podrá grabar un nuevo dato en el registro de entrada del port A. (El registro de entrada está vacío.)

Modalidad 2: Cuando ARDY es lógico 1, datos

válidos estarán preparados en

el registro de salida del port A.
Contrariamente a la modalidad 0, los datos se pondrán únicamente en el A0 hasta A7 después de haber activado el dispositivo periférico ASTB.

Modalidad 3: ARDY se pondrá constantemente a lógico 0.

B0 hasta B7 Este bus de 8 bits (bidireccional, Tristate) sirve para la comunicación entre el port B del PIO y el dispositivo periférico. Dependiendo de la modalida pueden intercambiarse informaciones de control, estado y datos.

BSTB (Port B Strobe) Esta entrada del port B corresponde en su función a la entrada ASTB, pero con una excepción: En la modalidad 2 se escribirán los datos en el port A.

BRDY (Register B Ready) Esta salida del port B corresponde en su función a la salida ARDY, pero con una excepción: En la modalidad 2 BRDY es lógico 1, cuando el port A puede recibir nuevos datos del dispositivo periférico.

Puesta a cero del PIO (reset)

En cada conexión de la alimentación de corriente el PIO se inicializa automáticamente con un estado definido, el cual puede tener el siguiente aspecto:

1.) Los dos registros de máscara del port se pondrán a cero para todos los bits de datos del port.

2.) Las líneas de datos del port se encontrarán en estado de alto ohmiaje.
3.) Ambos ports se encuentran en la modalidad 1 (modalidad de señal).
4.) Los registros de direccionamiento de vector no se han puesto a cero (éstos se aplicarán para apoyar la modalidad 2 del Z80-µP).
5.) Los dos flip-flop's de liberación de interrupt han inicializado a cero. El PIO está bloqueado para los interrupts.
6.) Los dos registros de salida del port se han puesto a cero.

En algunos casos de aplicación se deseará inicializar el PIO a cero durante el funcionamiento, a pesar de que el PIO, debido a su limitación de 40 pins, no posee una entrada especial de RESET. Sin embargo, mediante la aplicación de una brecha individual, AND puede conseguirse fácilmente. Para ello se conectan las entradas de la brecha con M1 y el RESET del Z80-µP y se conecta la salida del GATE AND en el M1 del PIO (pin 37). Siempre que se provoque un reset del sistema, se procuce también un reset del PIO.

El registro del vector de interrupt:

El Z80-PIO ha sido diseñado para la modalidad 2 del Z80-µP. Para ello posee un registro interno de vector de 8 bits, cuyo contenido, en el caso de una confirmación de interrupt por parte del Z80-µP, se envía al bus de datos del Z80. (Condición para ello es que el PIO tenga en este momento la mayor prioridad de todas las unidades, o sea, que permanezca disponible para el manejo de interrupts.) El Z80-µP forma con ello y con el contenido del registro de interrupt del Z80 un vector de

direccionamiento de 16 bits para la tabla de bifurcaciones de la memoria (véase IM2, capítulo 8). El Bit 0 del vector de interrupt deberá ser 0, para poder ser reconocido por el PIO como tal y cargarse en el registro interno del vector de interrupt (C/D SEL = lógico 1). La palabra de mando correspondiente para poder cargar el registro de vector de interrupt tiene el siguiente aspecto:

D7	D6 ....D1	D0	
V7	V6 ....V1	0	reconoce la palabra de control como vector del interrupt

La modalidad del registro de control:

El Z80-PIO dispone de cuatro distintas modalidades (tipos funcionales):

Modalidad 0 (Modalidad de salida)
Modalidad 1 (Modalidad de entrada)
Modalidad 2 (Modalidad bidireccional)
Modalidad 3 (Modalidad de control)

El port A puede gestionarse en cualquier modalidad. El port B puede, prescindiendo de la modalidad 2 e independientemente del port A, trabajar en las distintas modalidades. El código de control para cargar la modalidad del registro de control tiene el siguiente formato:

D7	D6	D5	D4	D3	D2	D1	D0
M1	M0	X	X	1	1	1	1

Los bits D3 hasta D0 del código de comando identifican a la misma como modalidad control. El contenido de los bits D5 y D4 es indiferente. El D7 y el D6 definen la modalidad en la que el port correspondiente deberá funcionar:

D7	D6	Modalidad
0	0	0
0	1	1
1	0	2
1	1	3

Unicamente cuando se programa la modalidad 3, el PIO interpreta el siguiente código de control como comando de entrada/salida. El estado lógico de los distintos bits define si las líneas del bus de datos del port actúan como entrada o salida. El código de control tiene el siguiente formato:

D7	D6	D5	D4	D3	D2	D1	D0
E/A7	E/A6	E/A5					E/A0

Si por ejemplo el bit D5 es lógico 1, la A5 trabajará como entrada (siempre y cuando el port A se haya cargado con el código de control).

El código de control del interrupt:

Formato:	D7	D6	D5	D4	D3	D2	D1	DO
	Liber. Inter.	AND/ OR	HIGH/ LOW	Másc. sigue	0	1	1	1

Bit 7: Con el bit 7 se determina si el port correspondiente puede generar interrupts o no (bit 7=0 : los interrupts están bloqueados, bit 7=1 : los Interrupts están liberados). Si durante el bloqueo de un interrupt aparece una condición de interrupt, el interrupt se efectuará posteriormente a la liberación.

Bits D3,D2,D1,DO: Estos 4 bits con el valor de 0111 definen el comando de control como palabra de comando de interrupt.

Bit D4: Este bit indica en la modalidad 3 si el próximo comando deberá interpretarse como máscara de bit (D4 = lógico 1). Si está activado (lógico 1), se borrarán en cada modalidad los interrupts previstos. (Una máscara de bit será requerida en la modalidad 3 cuando el port pueda generar interrupts o bien cuando durante el funcionamiento se requiera una nueva máscara de bit.)

Bit D5: Este bit únicamente actúa en la modalidad 3 e indica en que estado activo tendrá lugar la comparación entre la máscara de bit y los datos de entrada. (D5 = 1: high activo, D5 = 0: low activo.)

Bit D6: También este bit actúa únicamente en la modalidad 3 e indica con qué función lógica (AND/OR) tendrá lugar la comparación entre la máscara bit y los datos de entrada (D6 = 1: AND, D6 = 0: OR).

En la modalidad 3 no se provoca un interrupt mediante estados especiales de handshake, sino por estados especiales del bus de datos del port. En todas las modalidades el port correspondiente podrá ser liberado o bien bloqueado mediante el bit D7 del comando de interrupt. Si se desea controlar la liberación del interrupt sin tener que modificar en la modalidad 3 los bits D6, D5 y D4, disponemos del siguiente comando:

	D7	D6	D5	D4	D3	D2	D1	D0
Liber. Inter.	X	X	X	0	0	1	1	

Ejemplo: Comando = 83H: liberar interrupts
Comando = 03H: bloquear interrupts

Con este comando puede realizarse en cualquier modalidad la liberación de interrupts.

Cuando en la modalidad 3 esté activado el bit D4 del comando de interrupt, el siguiente comando será interpretada por el DIO como máscara de bit para ser cargada en el registro de máscaras bit. La máscara de bit tiene el siguiente formato:

D7	D6	D5	D4	D3	D2	D1	D0
MB7	MB6	MB5	MB4	MB3	MB2	MB1	MB0

Los bits de máscaras MB7 hasta MB0 están funcionalmente asignados a las líneas de datos Port A7 hasta A0 para el port A, B7 hasta B0 para el port B. Si un bit de máscaras es lógico 1, no se utilizará para la comparación (estará "sin máscara").

Modalidad 0 (modalidad de salida):

En esta modalidad la CPU podrá escribir en todo momento un dato en el registro de salida del port, donde quedará almacenado hasta ser sobrescrito por un nuevo dato. (Mediante un cambio en la modalidad 1 (modalidad de entrada) la CPU podrá volver a leer el dato grabado últimamente en el registro de salida.)

Aprovechando el sistema de acuse de recibo del handshake controlado por el interrupt entre el port PIO y el dispositivo periférico, el port creará siempre un Interrupt cuando el dispositivo periférico haya recibido el dato anterior y la CPU pueda escribir así un nuevo carácter en el registro de salida. Tanto el port A como el port B del PIO podrán administrarse independientemente uno de otro en la modalidad 0. El mecanismo del handshake se iniciará mediante una operación de grabación de la CPU en el registro de salida del port.

Modalidad 1 (modalidad de entrada):

En esta modalidad la CPU podrá leer el registro de en-

trada del port. Con la línea strobe podrán grabarse datos del dispositivo periférico al registro de entrada del Port PIO. Mientras no se utilice el sistema acuse de recibo del handshake, el strobe deberá inicializarse en lógico 0 para que los datos puedan ser aceptados del bus de datos del port en el registro de entrada.

Para conseguir una sincronización exacta de los ciclos de lectura del port de la CPU, con la entrada de datos del dispositivo periférico en el registro de entrada del port, se aplicarán las líneas del handshake. El port creará entonces un interrupt, siempre y cuando un nuevo dato esté disponible para ser leído en el registro de entrada del port para la CPU. El mecanismo del handshake se iniciará mediante una operación de lectura de la CPU. Tanto el port A como el port B podrán administrarse independientemente uno de otro en la modalidad 1.

Modalidad 2 (modalidad bidireccional):

En las modalidades 0 y 1 únicamente es posible la transferencia de datos (entre el port PIO y el dispositivo periférico) en una dirección. La modalidad 2 permite sin embargo una tranferencia de datos hacia ambas direcciones, es decir, una combinación de las modalidades 0 y 1 pero en la que únicamente podrá trabajar el port A.

Las líneas de handshake del port A sirven para la emisión de datos y las líneas de handshake del port B para el controñ de la entrada de datos. Contrariamente a la modalidad 0, en la modalidad 2 se envían los datos del registro de salida del port A en el port del bus de datos después de activar la línea ASTB. El port B deberá trabajar en la modalidad 3 cuando el port A se haya inicializado en modalidad 2. Además, la máscara de bit para el port 2 deberá estar ya activada en el FFH para que el port B no pueda distorsionar la transferencia del handshake del interrupt del port A. Para la transferen-

cia de datos asíncrono a través del port B queda únicamente la técnica de consulta (Polling).

Modalidad 3 (modalidad de control):

Esta modalidad es especialmente adecuada para funciones de control así como para controlar situaciones de estado de dispositivos periféricos. Contrariamente a las modalidades descritas hasta ahora, ésta no utiliza líneas de handshake (los registros de datos del port pueden ser escritos o leídos en cualquier momento) y no trabaja tampoco byte a byte, sino bit a bit, es decir, cada línea de datos del port puede programarse ya sea como línea de entrada o bien de salida. Con excepción de la modalidad 2 (para el port A), la modalidad 3 activa la correspondiente línea de ready a lógico 0. Los dos ports PIO pueden trabajar independientemente uno de otro en la modalidad 3. Se provoca un interrupt en esta modalidad si hay liberación de interrupts y los datos del bus de datos del port satisfacen la ecuación lógica definida a través de el código de control de interrupts y la máscara de bits. Supongamos que el bit D6 del código de control del interrupt es lógico 0 (función OR) y que el bit D5 del mismo es lógico 1 (high activo). Además las máscaras de bit son 00H, es decir, todas las 8 líneas de datos del port están incluidas en el control lógico de comparación. Cuando cualquier línea de datos adopta el estado high (lógico 1), entonces se habrá cumplido la ecuación OR y el port provocará un interrupt. Si como consecuencia otra línea de datos adopta el valor lógico 1, ésto no ocasionará un segundo interrupt porque el resultado de la ecuación ya no podrá ser más que verdadero.

En la modalidad 3 se produce el interrupt únicamente cuando la ecuación lógica se convierte en verdadera (es decir que antes era falsa).

Z80-PIO - Técnica de interrupt:

Como ya se ha comentado, el Z80-PIO soporta la modalidad 2 del interrupt de la CPU del Z80. Después de que el PIO haya enviado un interrupt al Z80-µP, éste lo confirmará desactivando al mismo tiempo las líneas M1 e IORQ. A continuación el PIO con la máxima prioridad (IEI = lógico 1, IEO = lógico 0) envía el vector de 8 bits del port interrumpt (port A es internamente de mayor prioridad del PIO al port B) al bus de datos de la CPU. Pueden administrarse hasta cuatro PIO's sin el hardware adicional en una daisy chain. La entrada IEI del PIO de mayor prioridad se conecta a +5V (a través de una resistencia). Mediante interrelaciones adecuadas AND podrán encadenarse más de 30 PIO's en una daisy chain. Para ello se utiliza la llamada técnica "Look Ahead".

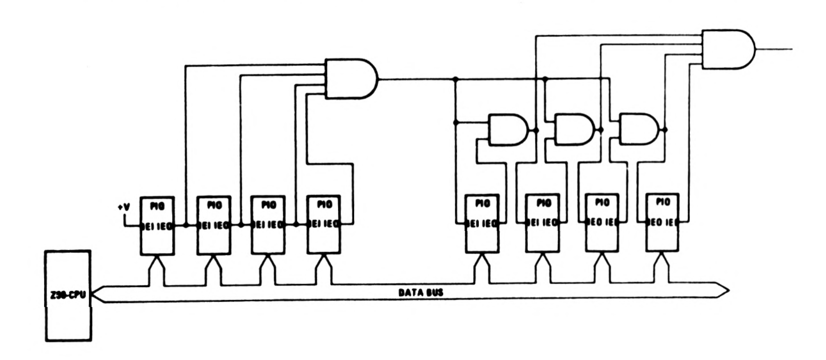


Fig. 5.7 Ampliación de la daisy chain mediante la técnica del "Look Ahead".

5.5 Ejemplo con el Z80-PIO:

Con el fin de poder hacer una demostración de la aplicación práctica de un PIO en las distintas modalidades, expondremos a continuación un control simplificado del proceso. En el esquema de bloques de control (Fig. 5.8) pueden distinguirse las diferentes unidades de funcionamiento. Cuatro sensores sirven para la lectura de valores de magnitudes relevantes del proceso (por ejemplo temperatura, presión, humedad, paso de masa, etc.) que finalmente se someten a una transformación y adaptación de señales (corrección de líneas de identificación, filtraje, conversión en señales de unidad, por ejemplo 0 hasta 20 mA, etc.).

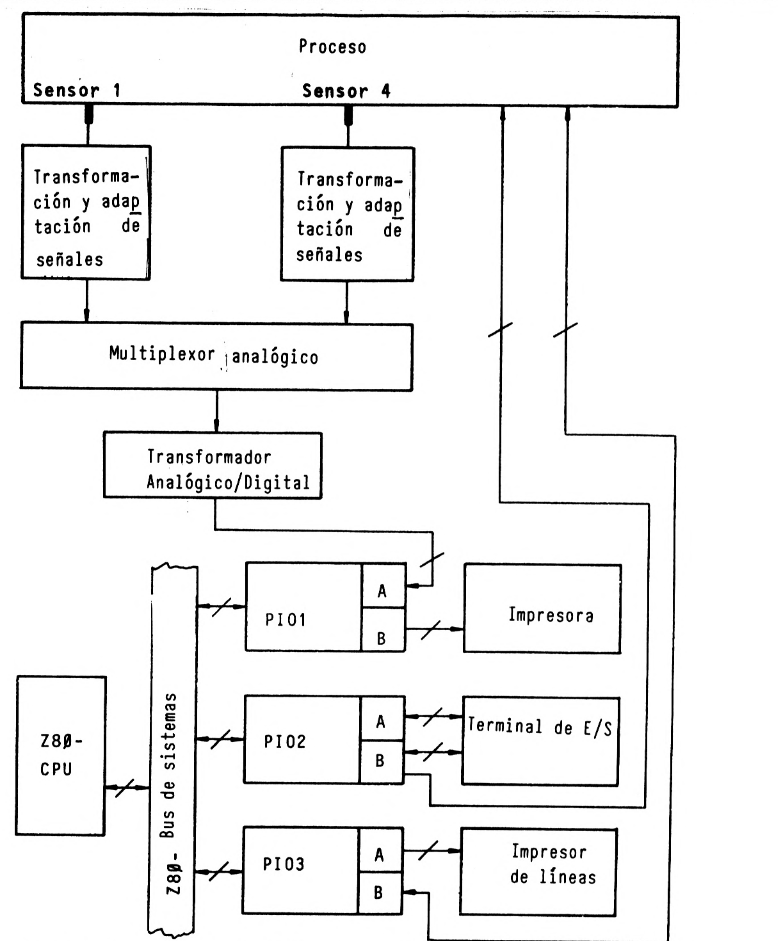


Fig. 5.8 Circuito de bloques del control de proceso.

Para no tener que agregar a cada sensor un propio ADU (conversor análogico digital), las señales de las unidades se multiplicarán mediante un multiplexor analógico para proporcionar seguidamente al ADU en una línea. Para simplificar las cosas partimos de la base de que el ADU deberá proporcionar una señal de salida de ocho bits (ADU de ocho bits). Cada ADU posee por lo menos las líneas de control SC (Start Conversion) y EOC (End Of Conversion) necesarias para el handshake. Para la aplicación en sistemas controlados por microprocesadores son adecuados los ADU's que trabajan según el principio de la aproximación paso a paso. Ofrecen una elevada velocidad, gran resolución y precio moderado. Unicamente en casos especiales deberán utilizarse bloques más caros, como ADU's integrados para resoluciones especialmente elevadas (por ejemplo de 16 bits) o bien ADU's paralelos para conversiones extremadamente rápidas (por ejemplo 100ns). El multiplexor analógico puede, o bien controlarse a través de un contador módulo de 4 (conmutación secuencial de posiciones de medición), o mediante el bus de direcciones del Z80 (posibilidad de libre control). El port A del PIO 1 recibe los datos del ADU, y por lo tanto debe estar programado en modalidad de entrada. En el port B del PIO 1 se ha conectado una impresora, que realiza los protocolos del proceso.

El port A del PIO 2 alimenta una terminal de entrada/salida en la modalidad bidireccional. El port B del PIO 2 se encuentra necesariamente en la modalidad de control y alimenta el proceso con ocho líneas de control, a través de las cuales se accionan diversos elementos de ajuste (válvulas, cursores, válvulas de mariposa, relés, motores, embragues, etc.).

Una impresora de líneas conectada al port A del PIO 3

protocoliza datos importantes de forma análoga.

El port B del PIO 3 trabaja nuevamente en la modalidad 3; sin embargo, el port, en comparación con el port B del PIO 2, es capaz de crear interrupts. Las líneas de datos B7 hasta B5 están programadas como entradas y reciben mensajes de alarma del sistema de proceso (por ejemplo, mensaje de caída de red, mensaje elevación de temperatura, mensaje de sobrepresión). Cuando por lo menos una de estas entradas es lógico 1, el port genera un interrupt. B4 hasta B0 son salidas y controlan elementos de ajuste del proceso.

Ya que el sistema contiene pocos componentes de E/S, el direccionamiento directo del PIO será posible mediante líneas de direcciones (véase Fig. 5.9).

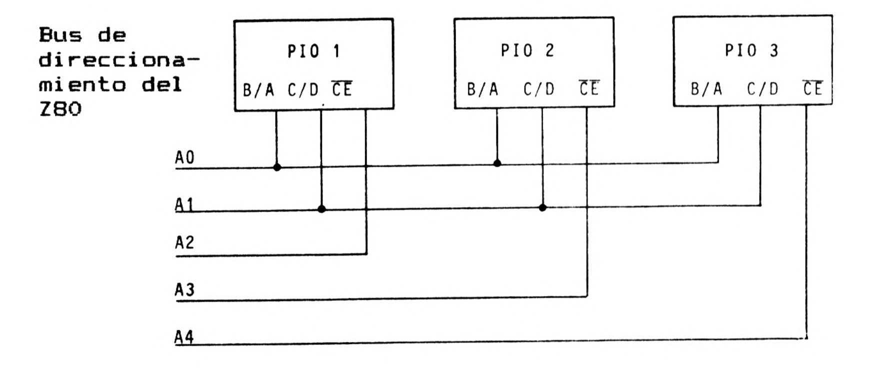


Fig. 5.9 Decodificación lineal de direccionamiento del PIO.

Para los PIO's son válidos:

A0	A1	Port seleccionado
0	0	A Datos
0	1	B Datos
1	0	A Comando de control
1	1	B Comando de control

Los distintos PIO's serán seleccionados mediante las siguientes direcciones:

A7	A6	A5	A4	A3	A2	A1	A0	PIO
X	X	X	1	1	0	X	X	1
X	X	X	1	0	1	X	X	2
X	X	X	0	1	1	X	X	3

Para que los PIO's puedan ser gestionados en la forma debida, tendrán que ser previamente programados. Esto ocurre siempre durante la llamada fase de inicialización del sistema, pero también durante el proceso del programa cuando debe ser modificada la modalidad. La siguiente rutina de inicialización describe la aplicación de los PIO's para el caso de utilización citado. Ya que el A7, el A6 y el A5 no están decodificados en el direccionamiento del PIO, resultan para los PIO's un gran número de posibles direcciones.

El siguiente programa utiliza las direcciones del PIO que se detallan a continuación:

D7	D6	D5	D4	D3	D2	D1	D0	PIO
0	0	0	1	1	0	X	X	1
0	0	0	1	0	1	X	X	2
0	0	0	0	1	1	X	X	3

Rutina de inicialización del PIO:

Bloquear todos los interrupts del PIO:

```
DI                  ; Bloquear interrupts del Z80
LD A,03H            ;
OUT (1AH),A         ; Bloquear PIO1/interrupt port A
OUT (1BH),A         ; Bloquear PIO1/interrupt port B
OUT (16H),A         ; Bloquear PIO2/interrupt port A
OUT (17H),A         ; Bloquear PIO2/interrupt port B
OUT (0EH),A         ; Bloquear PIO3/interrupt port A
OUT (0FH),A         ; Bloquear PIO3/interrupt port B
EI                  ; Liberar interrupt Z80
```

PIO 1 / Port A en modalidad de entrada:

```
LD A,00H            ;
OUT (1AH),A         ; Vector de interrupt port A
LD A,4FH
OUT (1AH),A         ; Modalidad 1 - comando de control
```

PIO 1 / Port B en modalidad de salida:

```
LD A,02H
OUT (1BH),A        ; Vector de interrupt port B
LD A,0FH
OUT (1BH),A        ; Modalidad 1 - comando de control
```

PIO 2 / Port A en modalidad bidireccional:

```
LD A,04H
OUT (16H),A        ; Sal. del vector de inter. port A
LD A,06H
OUT (17H),A        ; Ent. del vector de inter. port A
LD A,8FH
OUT (16H),A        ; Modalidad 2 - comando de control
```

PIO 2 / Port B en modalidad de control

```
LD A,FFH
OUT (17H),A        ; Modalidad 3 - comando de control
LD A,00H
OUT (17H),A        ; Registro de E/S: Salida
LD A,17H
OUT (17H),A        ; Sigue máscara
LD A,FFH
OUT (17H),A        ; Todos los bits con máscara
```

PIO 3 / Port A en modalidad de salida:

```
LD A,08H
OUT (0EH),A        ; Vector de interrupt port A
LD A.0FH
OUT (0EH),A        ; Modalidad 0 - comando de control
```

PIO 3 / Port B en modalidad de control:

```
LD A,0AH
OUT (0FH),A        ; Vector de interrupt port B
LD A,FFH
OUT (0FH),A        ; Modalidad 3 - comando de control
LD A,E0H
OUT (0FH),A        ; B7 hasta B5 entradas,
                     B4 hasta B0 salidas
LD A,37H
OUT (0FH),A        ; OR, high activo, sigue máscara
LD A,1FH
OUT (0FH),A        ; B7 hasta B5 serán controladas
```

Liberar todos los interrupts del PIO:

```
LD A,83H
OUT (1AH),A        ; Liberar PIO 1/interrupt port A
OUT (1BH),A        ; Liberar PIO 1/interrupt port B
OUT (16H),A        ; Liberar PIO 2/interrupt port A
OUT (17H),A        ; Liberar PIO 2/interrupt port B
OUT (0EH),A        ; Liberar PIO 3/interrupt port A
OUT (0FH),A        ; Liberar PIO 3/interrupt port B
```

El bloqueo de los interrupts del PIO al principio de la rutina de inicialización tiene como objeto eliminar los interrupts no deseados durante esta fase.

A los ports de este programa se les han asignado los siguientes vectores de interrupt:

PIO	Port	Vector
1	A	00H
1	B	02H
2	A-Salida	04H
2	A-Entrada	06H
3	A	08H
3	B	0AH

Supongamos que hemos cargado el registro del interrupt del Z80-µP con 80H; entonces se encontraría nuestra tabla de bifurcaciones de interrupt en el campo de memoria 8000H hasta 800AH. Los seis vectores del PIO intencionadamente tienen una distancia mínima de 2 bits, porque la tabla de bifurcaciones contiene direcciones completas de 16 bits que apuntan luego a las rutinas de control de interrupts correspondientes. Supongamos que la rutina de control para el port A del PIO 1 empieza, por ejemplo, desde la dirección 9000H. La dirección de memoria 8000H deberá contener entonces el dato 00H y la dirección 8001H el dato 90H.

La prioridad de los PIO's entre si se efectuará automáticamente en la disposición de daisy chain. Los dos Ports de un PIO tienen prioridad interna a través de una daisy chain (el port A tiene prioridad ante el port B).

Aquí también es importante cerrar las rutinas de control de interrupt mediante la instrucción especial de retroceso RETI. Ya que el Z80-µP después de la aceptación de un Interrupt queda bloqueado para otros interrupts, éstos deberán volverse a liberar inmediatamente antes de la instrucción de retroceso mediante la instrucción EI. Si se permiten nuevos interrupts antes de la instrucción de retroceso (EI), podrán resultar interrupts encadenados.

6. Transmisión de datos en serie

6.1 Generalidades:

Cuando deben transmitirse datos a través de grandes distancias, es muy laborioso enviar el bus de datos de 8 bits de ancho de forma paralela a través de líneas. Por este motivo se transforma la información de 8 bits en una de tipo serie, donde siempre se envía un bit por vez, mediante componentes integrados (componente de comunicación de serie).

Si por ejemplo y por motivos de espacio se desean ahorrar componentes y a su vez las exigencias de velocidad no son demasiado elevadas, podrá realizarse una transformación mediante el correspondiente programa de software. Esto podría realizarse de tal modo que para la transmisión de datos únicamente se utilizará el DO y los datos serán leídos del acumulador, con lo cual, si aplicamos las instrucciones de rotación aparecerán en el DO bit a bit. Un bucle de tiempo incorporado en el programa determinará la velocidad de transmisión. de forma análoga podrán leerse datos del DO y ser transformados en su configuración inicial de ocho bits. Mediante este método, el procesador estará siempre ocupado con los procesos de transformación y no podrá aceptar ninguna tarea. Sin embargo, si aplicamos componentes de comunicación en serie, el procesador podrá realizar otros procedimientos durante el tiempo de la transformación de datos, ya sea de paralelo a transformación en serie y viceversa; por ejemplo podrá realizar un test de razonabilidad de los datos leídos o bien manejar otras unidades periféricas siempre que dispongan de dispositivos de interrupt. En todo caso se ahorran líneas para la transmisión.

Cuando se trata de comunicación Simplex bastará una línea (naturalmente con la masa) para la comunicación en

serie siempre y cuando el emisor y el receptor estén sintonizados. Sin embargo, si el receptor puede leer más de prisa de lo que pueda emitir el emisor, se necesitarán líneas de control adicionales.

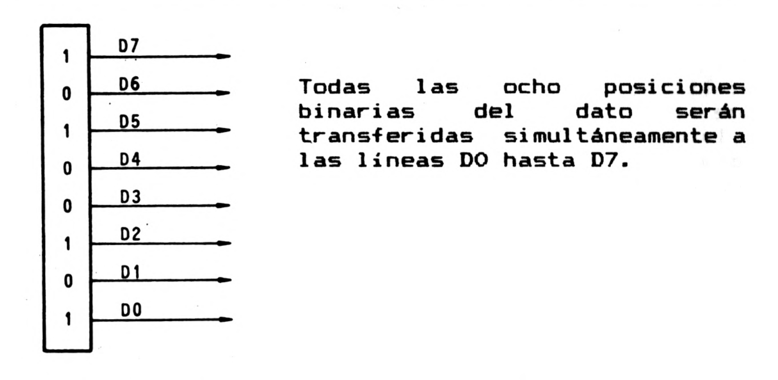
Aquí distinguimos entre la transmisión de datos en serie síncrona y asíncrona. La transmisión de datos síncrona trabaja con pilas de varios centenares de bits, cuyo inicio y fin están provistos con bytes de sincronización. Los caracteres de sincronización sirven para sincronizar el emisor y el receptor. La transmisión de datos asíncrona posibilita únicamente la sincronización del emisor y receptor para la duración de un carácter (dato). El inicio y fin de un carácter está determinado mediante bits de inicio y fin (Start/Stop). Ya que la técnica asíncrona de transmisión tiene pocas exigencias en las instalaciones de transmisión, es más barata y por ello se aplica más a menudo. Sin embargo el tiempo invertido es mucho más elevado que en la transmisión síncrona de datos. Si por lo tanto las exigencias de tiempo son muy elevadas (por encima de los 20 kBd), deberá pasarse a la transmisión síncrona.

Ejemplos de aplicación para transmisiones de datos en serie son, por ejemplo, la conexión de estaciones de Floppy-Disk, impresoras, terminales, modems, etc., a ordenadores así como el proceso de teletratamiento de datos, proceso de control a distancia, etc.

Si deben salvarse grandes distancias, se aplicarán los llamados modems. Con la ayuda de los mismos podrán dirigirse informaciones digitales, tales como a través de la red telefónica pública. Esto es muy importante en

el llamado proceso de teletratamiento de datos, en el que existe una distancia de varios centenares de kilómetros entre el centro de cálculo y las distintas estaciones de entrada y salida. Pero también en el campo industrial, por ejemplo en las grandes industrias del acero o bien refinerías, pueden separar varios kilómetros el lugar de medición del ordenador central.

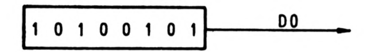
Valores de medición también pueden transmitirse de forma análoga (corrientes memorizadas, transmisión de señales de frecuencia analógica), pero la transmisión digital tiene muchas más ventajas cuando se trata de grandes distancias. En primer lugar en las transmisiones digitales pueden transmitirse valores de medición y cifras de cualquier número de posiciones. Una distorsión de los datos transmitidos a través del medio de transmisión (modificaciones de las resistencias de la línea, resistencias del aislamiento, capacidades del cable, distorsiones de la corriente, etc.) apenas puede tener lugar, a no ser que las influencias de distorsión puedan ser tan grandes que las señales transmitidas sean completamente borradas o bien que por distorsiones de la corriente muy grandes aparezcan impulsos que no hayan sido transmitidos. Incluso si aparecieran tales fallos de transmisión (lo que ocurre muy rara vez), éstos serían detectados mediante métodos adecuados de verificación (bits de control, test del CRC, etc). En segundo lugar pueden aplicarse para la transmisión cables sencillos de teléfono y de telex, pudiendo así transmitirse datos a través de grandes y mayores distancias a un precio relativamente asequible.



Todas las ocho posiciones binarias del dato serán transferidas simultáneamente a las líneas DO hasta D7.

Fig. 6.1 Transferencia en paralelo del dato A5H.

Ejemplo: Transferencia en serie del dato A5H.



Las ocho posiciones binarias del dato serán transferidas en una línea (aquí: DO) en secuencia temporal.

Ejemplo: Transformación del software paralelo / serie.

Al llamar este subprograma, el dato a ser transferido deberá encontrarse en el registro C. El dato se enviará al DO del port 1.

```
SEROUT:    LD B,8
           LD A,C
NEXT:      OUT 1
           RRA
           DJNZ NEXT
           RET
```

Descripción del programa:

El registro B será cargado con el número de bits a transmitir. A continuación se cargará con LD A,C el dato a transmitir en el acumulador. Después empezará el bucle NEXT. Se transfiere el acumulador al port 1, del que únicamente se habrá conectado el DO. En el DO se encontrará pues el bit 0 del dato. Con RRA se desplazará el contenido del acumulador en una posición hacia la derecha, de modo que el bit 1 del dato se encontrará en el DO del acumulador. La línea DJNZ NEXT decrementa el contador del bucle (registro B), retrocediendo el programa a NEXT: OUT1, mientras B sea desigual a 0. Mediante esta nueva transmisión del acumulador al port 1 dependerá el DO del bit 1 del port del dato. El bucle se repetirá hasta que los 8 bits del dato se hayan transmitido al DO del port 1 (B=0).

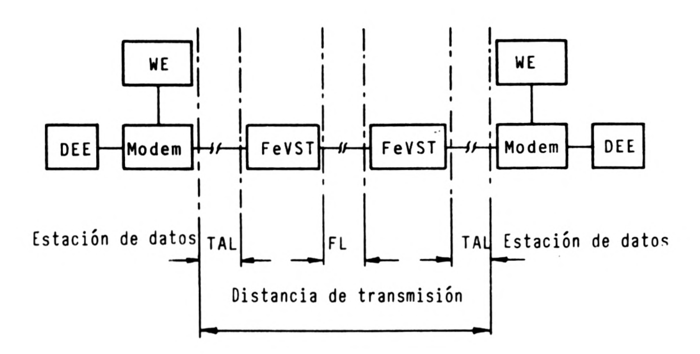
Los distintos bits deberían transferirse con una cuota determinada de transmisión y además estar provistos para el caso de tener que hacer una transmisión asíncrona, con bits de inicio, de fin y eventualmente con bits de paridad. Para aquella persona que haya comprendido el principio de la transformación en el programa ejemplo citado, no le será difícil preparar el programa requerido para el caso específico de aplicación. A pesar de ello, indicaremos como estímulo un programa para la transmisión de un dato a través de un teletipo con 110 Bd. El dato a transmitir deberá encontrarse en el registro C; el registro B contiene el número de bits a ser transferidos y el carryflag se utilizará en relación con las instrucciones de rotación para la interposición de un bit de inicio y dos bits de final.

```
SEROUT:     LD B,11
            LD A,C
            ORA              ; Inicial. con cero carryflag
            RLA              ; Bit inicial en el accu A(1)
NEXT:       OUT 1            ; Emisión al port 1
            CALL RETR        ; 9 ms de retardo
            RRA              ; Próximo bit según A (1)
            SCF              ; Carry contiene bit de final
            DJNZ NEXT        ; Nuevamente el bucle, en el
                               caso de que B no sea cero
            RET
```

Bucle de retardo de 9 ms.

```
RETR:       LD D,6
RT 1:       LD E,2000
RT 2:       DCR E            ; Bucle interno de espera de
                               1,5 ms
            JNZ RT2
            DCRD
            JNZ RT1          ; Bucle externo de espera
```

El retardo de 9 ms se consigue mediante dos bucles de retardo encadenados. El bucle interno de espera (1,5 ms) se repetirá seis veces, de modo que en total resultará un retardo de 9 ms. 9 ms por bit corresponde a una "Baud Rate" de 110. De modo muy similar puede leerse y reconstruirse un carácter que viene en serie, rotando el Accu hacia la izquierda, en lugar de hacia la derecha.



DEE: Organización de datos
WE: Dispositivo de selección
Modem: Modulador / demodulador
TAL: Línea de conexión de abonado
FL: Línea interurbana
FeVST: Central telefónica para conferencias

Fig. 6.2 Transmisión de datos de la red telefónica pública en la RFA.

En los modems con acoplamientos acústicos puede crearse el enlace (proceso como en las llamadas telefónicas usuales) mediante el disco de selección (resp. teclado).

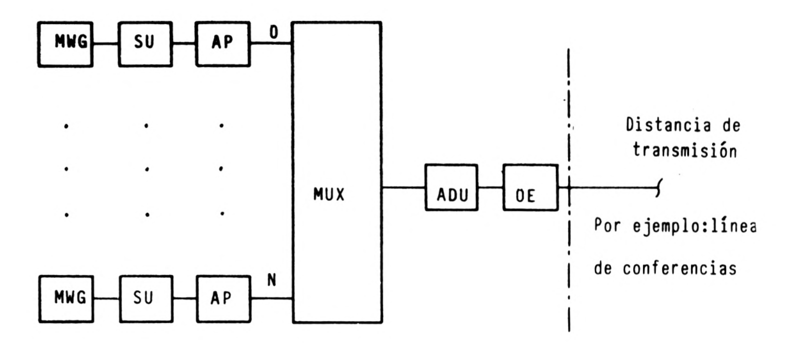


Fig. 6.3 Medición digital a distancia.

TVM: Transmisor de los valores medidos. Convierte un tamaño físico en una señal eléctrica definida. Para ello se dispone de una amplia gama de sensores (por ejemplo bandas extensométricas).

SP: Supresión de perturbaciones. Mediante filtros adecuados se filtrarán los ámbitos de frecuencia no utilizados con el fin de eliminar las perturbaciones que puedan existir.

AD: Adaptación. Según las propiedades del sensor deberán alinearse sus curvas características, reforzarse las señales eléctricas y limitarlas a

los límites superiores fijados. Además se realizará una separación potencial galvánica mediante transmisores o bien optoadaptadores.

MUX: Multiplexor. El multiplexor analógico sirve de conmutador del punto de medición. Este transmite en secuencia las señales que provienen del AD al TAD común.

ADU: Conversor analógico/digital. El MUX suministra al TAD señales analógicas de corriente que se mueven en campos definidos.

Ejemplos ESTANDARD:

0...+5 mA
-5....0...+5 mA
4...+20mA
-20...0...+20mA

El ADU convierte la señal analógica en una información adecuada digital y la transmite paralelamente a la dirección de transmisión.

IT: Instalación de transmisión. Esta convierte la información paralela en un formato de serie y transforma mediante modulación los impulsos de la corriente contínua en una señal de corriente alterna.

En el otro extremo de la distancia de transferencia se envían al ordenador los valores de medición de forma digital, a través del correspondiente receptor.

6.2 Modos de transmisión de datos:

Es de suma importancia tener una visión general sobre los modos de transmisión de los que se dispone para transmitir datos en serie. Este aspecto se proporcionará a continuación. En primer lugar distinguiremos entre líneas propias y líneas ajenas.

a) Líneas propias:

Si las distancias a ser salvadas no son demasiado grandes, generalmente se utilizan las líneas propias. Estas pueden ser - según las exigencias de velocidad y contingente de errores - líneas normales, cables coaxiales o bien conductores de luz. Las líneas normales son más baratas y son suficientes para exigencias mínimas. Las líneas coaxiales están blindadas contra influencias eléctricas perturbadoras y poseen una impedancia definida, de modo que las velocidades elevadas de transmisión son posibles, con una cuota reducida de errores. Los conductores de luz son completamente insensibles contra influencias perturbadoras eléctricas y permiten velocidades elevadísimas de transmisión. También se aplican para velocidades de transmisión bajas y medianas cuando la línea debe pasar por una zona eléctrica y/o magnéticamente muy saturada.

b) Líneas ajenas:

En casi todos los casos en los que los datos deben ser transmitidos a través de tramos largos, se aconseja - aunque sólo sea por motivos económicos - utilizar líneas ajenas. En la República Federal Alemana son éstas, en

primer lugar, las redes públicas y las líneas punto a punto de los Correos Federales Alemanes.

Las redes públicas de la RFA son:

- Red de telex (red selectora de teletipos)
- Red de datos (red general selectora para la transmisión de noticias)
- Red telefónica (red telefónica selectora)

La característica de las redes públicas es que el usuario deberá utilizar las instalaciones y las líneas de la RFA para poder obtener la comunicación deseada con cualquier otro usuario.

Las líneas punto a punto de la RFA son:

- Líneas telegráficas cedidas
- Líneas telefónicas cedidas
- Líneas de banda ancha cedidas

La característica de las líneas punto a punto es que éstas están únicamente reservadas para determinados usuarios y que exclusivamente pueden ser utilizadas por éstos.

En general existen las siguientes premisas para la decisión sobre el modo de transmisión que deberá utilizarse:

a) Las líneas punto a punto cuestan mensualmente un al-

quiler determinado que depende de la distancia y de la velocidad máxima de transmisión.

b) Al utilizar redes esténdard, los costes se calculan según la distancia de las zonas y las unidades de tiempo.

La tabla 6.1 representa una ayuda orientativa para la selección de los modos de transmisión de redes públicas más favorables:

Para 25 Km	hasta caracteres/mes (código de 8 Bits)
Red de telex 50 Bd	5.10^4
Red de datos 200 Bd	$1,2.10^6$
Red telefónica estándard 1200 Bd	23.10^6
Línea telefónica punto a punto 1200 Bd	$92,5.10^6$
Línea de banda ancha 48 KHz (40800Bd)	$3,2.10^9$

Para 150 Km

Red de telex 50 Bd	5.10^4
Red de datos 200 Bd	$1.7.10^6$
Red telefónica estándard 1200 Bd	33.10^6
Línea telefónica punto a punto 1200 Bd	$92,5.10^6$
Línea de banda ancha 48 KHz (40800Bd)	$3,2.10^9$

Para 500 Km

Red de telex 50 Bd	5.10^4
Red de datos 200 Bd	760.10^3
Red telefónica estádard 1200 Bd	55.10^6
Línea telefónica punto a punto 1200 Bd	$92,5.10^6$
Línea de banda ancha 48 KHz (40800Bd)	$3,2.10^9$

Tabla 6.1: Modos económicos de transmisión por red pública que dependen de la cantidad de caracteres transmitidos mensualmente.

Los datos más importantes de los modos de transmisión de datos por red pública pueden obtenerse de la tabla 6.2.

Velocidad de transm. (Bd)	Modo de transm.	Método	Porcentaje de errores (probab. de error. Bit)
50	Línea telegr. p. a p.	sx,hx,dx	$2...10 \cdot 10^{-7}$
50	Red de telex	sx,hx	$1...10 \cdot 10^{-6}$
100	Línea telegr. p. a p.	sx,dx	$1...10 \cdot 10^{-7}$
200	Línea telegr. p. a p.	sx,dx	$1...10 \cdot 10^{-7}$
200	Red DATEX	sx,dx	$2....8 \cdot 10^{-6}$
200	Lín. telefón. p. a p.	sx,dx	$1...10 \cdot 10^{-7}$
200	Red telefón. estánd.	sx,hx,dx	$2...10 \cdot 10^{-6}$
600/1200	Red telefón. estánd.	sx,hx,dx	$1...10 \cdot 10^{-5}$
600/1200	Lín. telefón. p. a p.	sx,hx,dx	$1...10 \cdot 10^{-6}$
sobre 1200	Lín. telefón. p. a p.	sx,hx,dx	aprox. 10^{-6}
4800/40800	Lín. de banda ancha	sx,hx,dx	aprox. 10^{-6}

Tabla 6.2 Modos de transmisión por redpública de datos

Teniendo en cuenta los puntos de ayuda facilitados, no debería tener dificultad en poder elegir la línea adecuada para su caso particular.

Un caso especial importante para todo poseedor de un ordenador personal es la llamada red del "Datex P20". Se creó en 1980 y posibilita la participación en la red Datex P, a la que la mayoría de bancos de datos internacionales se encuentran conectados.

Lo esencial de la red de DATEX P es que los datos se recopilan y son transmitidos en "paquetes". La Central de Correos dispone de unas instalaciones que se encargan de "atar estos paquetes" automáticamente. Sobre este particular deberá solicitarse en la Central de Telecomunicaciones la llamada "identificación", mediante la cual podrá entrarse en la red del Datex P20. Los derechos de conexión son fácilmente costeables; la conexión normal del teléfono podrá ser utilizada. Los gastos para la transmisión de datos en largas distancias son con la ayuda de la red del Datex P20, mucho más reducidos que utilizando las líneas normales interurbanas, ya que mediante "el empaquetado" las líneas interurbanas son mucho más efectivas. La conexión a la red del Datex P20 se realiza para usuarios particulares a través de un Modem con acoplador acústico.

6.3 Modems:

Los modems (Modem=Modulador/Demodulador) son aparatos para la transformación de señales de datos (binarios) en una forma adecuada para la transmisión a través de líneas de telecomunicación, utilizando métodos de modulación. Las señales transmitidas en las líneas de telecomunicación son señales de corriente alterna sin parte de corriente contínua. Como métodos de modulación tenemos la modulación de amplitud, de frecuencia y de fases, resultando algunas particularidades debido a la señal binaria de datos. Así pues el ancho de banda exigido para la transmisión por frecuencia modulada (FM) es prácticamente igual que para la amplitud modulada (AM), ya que efectivamente sólo existen dos estados de frecuencia (correspondientes a la señal de datos). En este caso particular se habla también del "método de conversión de la frecuencia" o bien FSK (Frequency Shift Keying). La aplicación de la modulación de fases (PM),

por motivos técnicos, se realiza según el método diferencial (DPM). La información a ser transmitida se posiciona en el cambio de fase y no en la posición de fase. El método (DPM) es únicamente adecuado para señales isocrónicas y se adapta perfectamente para el aumento de la velocidad de transmisión al pasar de 4, 8 o más posiciones de fase. Cuando se trata de posiciones de fase n, podrán transmitirse al mismo tiempo en n bits durante un intérvalo de manipulación 1n. Con el número de posiciones de fase aumentan también las exigencias en las instalaciones de transmisión, de modo que normalmente bastan 8 posiciones de fase. que permiten triplicar la velocidad de transmisión en comparación con 2 posiciones de fase. Ya que las líneas de telecomunicación únicamente ofrecen un campo de transmisión de 300 hasta 3400 Hz, con el DPM no podrán transmitirse más de 4800 bits/s. Cuando se trate de más de 4800 bits/s deberá derivarse a cables especiales de banda ancha. Para poder limitar las posibilidades técnicas de realización mejorando asímismo la compatibilidad de los modems de distintos fabricantes, se han elaborado normas de estandarización: en Europa han sido fijados por la CCITT y en América por Bell.

Las normas más relevantes de la CCITT para los modems:

a) Modem paralelo (V.30)

Este modem permite velocidades de transmisión de 20 a 40 caracteres por segundo y utiliza como medio de transmisión la red pública de teléfonos. Dicha línea trabaja con FM multifono y permite una retroalimentación sencilla (420 Hz). Contrariamente a los modems restan-

tes, los caracteres no se transmiten en serie sino en paralelo (4 canales de frecuencia, cada uno de 3 grupos). Esto se realiza en forma de código de selección 1 de 4. Los modems paralelos se utilizan en lugares donde muchos emisores transmiten datos a un receptor central y no se exigen velocidades de transmisión demasiado elevadas. Los emisores trabajan generalmente con entrada directa mediante teclado, son sencillos y de realización económica. La tolerancia para las distintas frecuencias es de +- 5 Hz.

Plan de frecuencia:

	Canal			
Grupo	1	2	3	4
A	920	1000	1080	1160
B	1320	1400	1480	1560
C	1720	1800	1480	1960

Pueden construirse variantes con 15 (16), 63 ó 256 caracteres.

1.) Cantidad de caracteres 15 (16): Utilización de los grupos de frecuencia A y C con 20 C/s. La frecuencia correspondiente se emite a 25 ms de longitud, siguiendo a continuación una pausa de unos 25 ms. En lugar de la pausa pueden emitirse también el B2 y el B3 alternativamente. Después, la velocidad de la transmisión se aumenta a 40 C/s.

2.) Cantidad de caracteres 63: Utilización de los grupos de frecuencia A,B y C con 20 C/s.

3.) Cantidad de caracteres 256: Transmisión mediante dos semicaracteres, donde B3 caracteriza la mitad inferior y B2 la mitad superior del carácter (cuando la mitad inferior es la primera que se emite).

b) Modem para 200 Bd (V.21)

Este Modem permite velocidades de transmisión de 200Bd y utiliza como medio de transmisión la red pública de teléfonos o bien la red de datos. Esta trabaja asíncronamente, utiliza la FM y permite la modalidad duplex completo. El modem utiliza dos canales. Por el canal 1 se transmite y por el 2 se recibe.

Canal 1: lógico 0.....1180 Hz
lógico 1......980 Hz

Canal 2: lógico 0.....1850 Hz
lógico 1.....1650 Hz

La tolerancia de frecuencia es de +- 6 Hz.

c) Modem para 600/1200 Bd (V.23)

Este modem permite velocidades de transmisión de 600, ó 1200 Bd seleccionables. El canal de retorno creado para las señales de control y de acuse de recibo se ha interpretado para 75 Bd. Tanto la red pública de teléfonos como también la red de datos pueden utilizarse como medio de transmisión. Trabaja asíncronamente y utiliza FM, permitiendo únicamente una modalidad duplex

semicompleto.

Modalidad 1 (600 Bd) : lógico 0.....1700 Hz
lógico 1.....1300 Hz

Modalidad 2 (1200 Bd) : lógico 0.....2100 Hz
lógico 1.....1300 Hz

Canal de retorno (75 Bd): lógico 0......450 Hz
lógico 1......390 Hz

Las tolerancias de las frecuencias para las señales de los datos son de +- 10 Hz. Casi todos los Modems con acopladores acústicos que actualmente se ofrecen en el campo de los ordenadores personales, se ajustan a las normas CCITT V.21, o a las normas correspondientes de Bell (Bell 202).

d) Modem para 2400 Bd (V.26)

Este Modem permite velocidades de transmisión de 2400 Bd y utiliza predominantemente como medio de transmisión las líneas de alquiler. Trabaja sincrónicamente, utiliza DPM con modulación de 4 fases y una velocidad de pulsacióm de 1200 por segundo.

e) Modem para 4800 Bd (V.27)

Este modem permite velocidades de transmisión de 4800 Bd y utiliza como medio de transmisión líneas de alquiler. Trabaja sincrónicamente, utiliza DPM con modulación de 8 fases y una velocidad de pulsación de 1600 por segundo.

f) Modem para 48 kBd (v.35)

Este modem permite velocidades de transmisión de 48000 Bd y utiliza como medio de transmisión conexiones de banda ancha. Trabaja según selección, sincrónica o asincrónicamente y utiliza una banda lateral residual - modulación de amplitud.

Frecuencia portadora: 100 kHz +- 2 kHz

Velocidad de trans. síncrona: 1) 48000 bits/s +- 1 bits/s
2) 40800 bits/s +- 1 bits/s

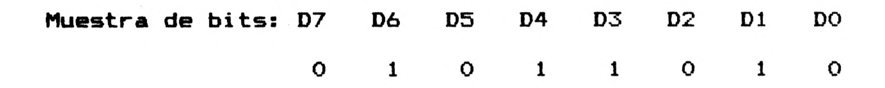
Velocidad de trans. asíncrona: entre 21 µs y 200 ms por Bit.

6.4 RS 232 C

El estándar RS232C es el estándar asincrono más difundido actualmente y describe tanto las condiciones eléctricas como las físicas para una transmisión de datos asíncrona, en serie de bits. La particularidad de una transmisión asíncrona de datos es que el flujo de datos no contiene ninguna información de cadencia (signo de sincronización), según el cual la estación receptora pueda "sincronizarse"; los datos son más bien transportados en unidades breves, equipadas con Bits de arranque y de paro. Ya que la cadencia del emisor y del receptor está por lo menos sobre el factor 16 y más elevada que la velocidad de transmisión pertinente, dichas unidades podrán seguramente ser transmitidas siempre y cuando no ejerzan una influencia perturbadora demasiado grande en el medio de transmisión. Ya que el microprocesador Z80 - como todos los demás procesadores de 8 bits - procesa los datos byte por byte, cuando se trata de transmisiones asíncronas se transfieren los 8 bits de datos

secuencialmente.

A pesar de ello pueden utilizarse formatos más cortos, por ejemplo, un formato de 5 bits. Esto puede tener sentido, si los datos a ser transmitidos están codificados con menos de 8 bits (por ejemplo 5 bits - Datos de un conversor analógico de 5 bits a un conversor digital), ya que debido al número reducido de bits a ser transmitidos en serie aumenta la velocidad de los caracteres (número de los caracteres transmitidos por segundo), manteniendo la misma velocidad de Baud (número de bits transmitidos por segundo). El componente receptor inicializa entonces los bits restantes a cero, pudiendo así disponer los datos para el procesador en el formato usual de los bytes (conjunto de 8 posiciones binarias).



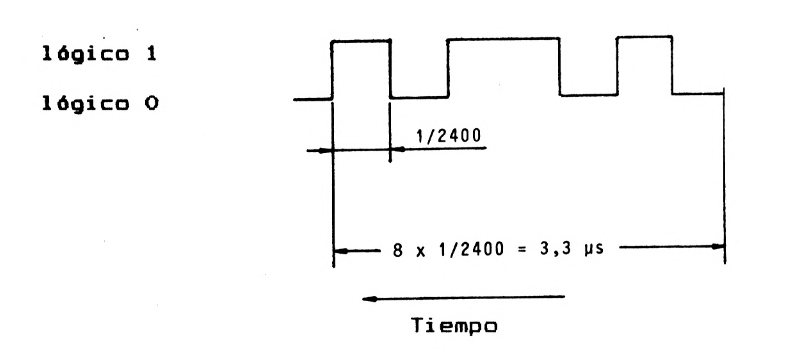


Fig. 6.4 Transmisión en serie del dato 5AH

Este ejemplo muestra la transmisión de bits en serie del dato 5AH con una velocidad de transmisión de 2400 Bd (Baud). Una velocidad de Baud de 2400 que posibilita la transmisión de 2400 bits por segundo.

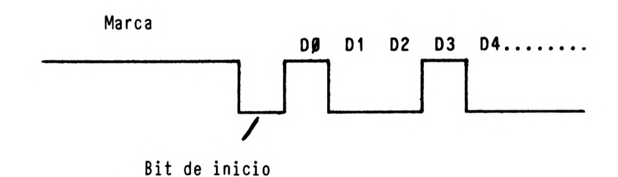
La unidad de medida de Baud (Bd), equivalente con bit/s, se utiliza principalmente en la técnica de transmisión en serie de bits, mientras que la unidad carácter/segundo (C/s) se ofrece para la técnica de transmisión en paralelo de bits, ya que para cada paso de transmisión se envía un carácter completo. Si en el primer caso de la transmisión en serie de bits desea saberse cuántos caracteres se emitirán por segundo, entonces se dividirá la velocidad de Baud por el número de bits requeridos para cada carácter. En nuestro ejemplo corresponderia para una velocidad de transmisión de 2400 Bd = 2400 bits/s a una velocidad de caracteres de 2400 : 8 = 300 C/s. Este cálculo no sería correcto para una transmisión asíncrona, ya que ésta no tiene en cuenta los bits de arranque, los de paro y eventualmente aquéllos utilizados para la paridad. Estos deberán naturalmente añadirse al divisor, de modo que al utilizar tan sólo un bit de arranque y uno de paro se consigue una velocidad de caracteres de 2400 : 10 = 240 C/s.

Las velocidades de transmisión usuales para el movimiento asíncrono en serie de bits son:

50 Bd	300 Bd	4800 Bd
75 Bd	600 Bd	9600 Bd
110 Bd	1200 Bd	19200 Bd
150 Bd	2400 Bd	

A continuación se estudiará detalladamente el formato de transmisión de 10 hasta 12 bits teniendo como base un dato de 8 bits:

a) Marca: El estado en reposo en la línea de transmisión corresponde a un uno lógico.



Mientras el emisor no transmita nada, pondrá una marca en la línea (Marking). El receptor esperará a la llegada de un bit de inicio, es decir, a la modificación del estado de la línea de lógico 1 a lógico 0. Una vez se haya transmitido el carácter y no le siga de inmediato otro, el emisor pondrá de nuevo una señal (lógico 1) en la línea.

b) Bit de arranque: El bit de arranque tendrá como marca el nivel opuesto, es decir, lógico 0. Indicará al receptor el inicio de una transmisión y motivará la lectura de los siguientes bits como datos.

c) Bit de paridad: Como continuación a los bits de datos puede seguir opcionalmente un bit de paridad. El estado lógico del bit de paridad depende de la paridad elegida (odd/even) y del dato asignado. Cuando se trata de una paridad par (even), entonces el bit de paridad será idéntico a lógico 1 cuando el número de unos en el dato sea impar.

Ejemplo: Dato = 54H, P = 1.

Cuando se trate de una paridad impar, el bit de paridad será idéntico a lógico 1 cuando el número de unos en el dato sea par.

Ejemplo: Dato = 55H, P = 1.

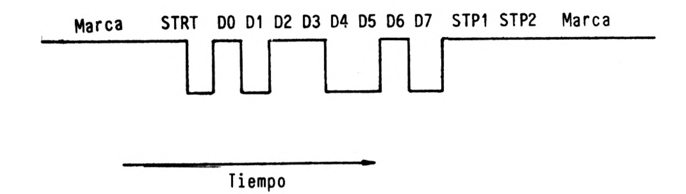
A pesar de que el bit de paridad no deberá ser aplicado, casi siempre se utiliza porque de este modo la distancia Hamming de cada código de transmisión será aumentada en 1, siendo así por lo menos de 2. De este modo podrán identificarse con seguridad los fallos de bits individuales y reducirse los porcentajes de errores en la magnitud de $p' = p^2$. La velocidad de error p de una línea pública de teléfonos es, por ejemplo, de $2...10.10^{-6}$. Si a través de una de estas líneas se envía información asegurada con bits de paridad de la forma descrita anteriormente, la posibilidad de errores (cuota de

errores residuales) será únicamente de $4.10^{-12}....1.10^{-10}$. En casos desfavorables aparece únicamente cada 10 millares de bits un error. Esta elevada seguridad de transmisión confirma en la mayoría de los casos la aplicación de bits de paridad, incluso aunque la velocidad de los caracteres quede algo reducida.

El receptor realiza asímismo un test de paridad. En el caso de detectar un error, se exigirá de nuevo el dato para volverlo a transmitir.

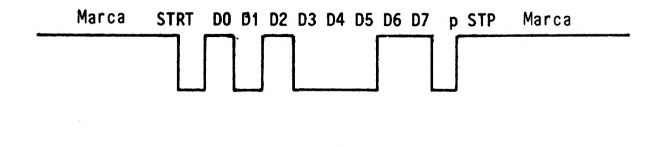
d) Bit de paro: El receptor espera al final de una transmisión de caracteres un bit de paro. Asímismo se considran como estándar 1 1/2 y 2 bits de paro. En la mayoría de los componentes de comunicación asíncrona puede seleccionar dentro del número de pasos de paro (1, 1 1/2 o bien 2). El bit de paro, o bien los bits de paro, tienen, igual que la marca, el nivel de lógica 1. A continuación del bit de paro (último) puede seguir ya sea un nuevo bit de inicio (lógico 0) o bien una señal (nivel constante 1).

Ejemplo: Formato completo de transmisión sin bit de paridad:



Esto es un ejemplo para un formato de transmisión asíncrono en serie de bits con un bit de inicio, 8 bits de datos y dos de paro. Como podrá apreciarse, se emite en primer lugar el bit de datos DO (LSB) de menor valor.

Ejemplo: Formato completo de transmisión con bit de paridad:



Tiempo

Esto es un ejemplo para un formato de transmisión asincrono en serie de bits con un bit de inicio, ocho

bits de datos, un bit de paridad y un bit de paro.

Conversión de paralelo/en serie:

Los microprocesadores procesan varios bits en paralelo; por ejemplo el Z80 procesa 8 bits simultáneamente. Para poder llevar los datos disponibles de forma paralela al formato en serie se cargarán los 8 bits en un registro de desplazamiento para desplazarlos después bit a bit del registro mencionado - ésto se realizará hasta que los 8 bits hayan sido "alejados" del registro; después se cargará el próximo byte en el registro de desplazamiento, etc. Los componentes de comunicación en serie f+r de los microprocesadores contienen uno de estos equipos de conversión e intercalan, después de haberse previamente programado, los bits de inicio, de paro y de paridad y los alejan de la corriente de datos que va llegando.

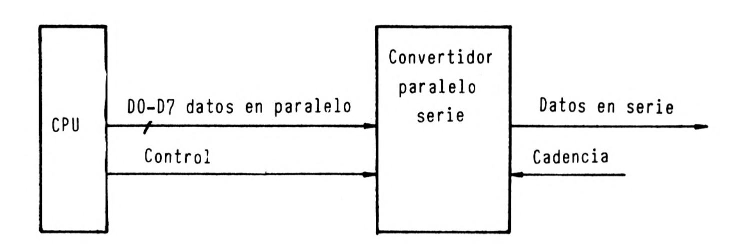


Fig. 6.5 Principio de la conversión de paralelo/serie

El RS232C como estándar especial asíncrono define en las líneas los siguientes niveles:

Lógico 1 : +3V hasta + 15V (=por ejemplo marca)
Lógico 0 : -3V hasta - 15V

Los niveles estándar del RS232C son ± 12 Voltios.

La corriente máxima de cortocircuito está limitada a 100 mA, de modo que el incendio de las líneas queda prácticamente descartado.

El estándar permite velocidades de transmisión hasta 20 K-Bits/s y una línea máxima de transmisión de 15 m. Ya que los componentes de comunicación en serie suministran niveles de TTL, deberán aplicarse conversores de nivel de TTL a RS232C y viceversa. Aparte de las realizaciones discretas con transistores podrán asimismo aplicarse circuitos especiales integrados para esta finalidad, como por ejemplo el MC1488 como amplificador de líneas y el MC1489 como receptor de líneas. Cada uno de estos componentes contiene cuatro receptores/emisores.

Los datos se transmiten en lógica negativa y las señales de control en lógica positiva.

El conector estándar RS232C requiere 25 polos de conexión. Para ello se aplican los llamados conectores miniatura D. El estándar determina las 25 conexiones, pero en la práctica se utilizan únicamente: 1, 2, 3, 4, 5, 6, 7, 8 y 20. Las distintas conexiones tienen la siguiente función:

Pin 1	:	Masa protectora de los dispositivos.
Pin 2	:	Transmitted Data (to communication equipment). Esta conexión es la fuente de datos desde la que se alimenta el dispositivo de comunicación. En la línea TxD se emitirán los datos.
Pin 3 (RxD)	:	Received Data (from communication equipment). Esta conexión es el fondo de datos que se alimenta del dispositivo de comunicación. En la línea RxD se recibirán los datos.
Pin 4 (RTS)	:	Request To Send (to communication equipment). A través de la línea RTS se le comunicará al dispositivo de comunicación la disponibilidad de transmisión.
Pin 5 (CTS)	:	Clear To Send (from communication equipment). A través de la línea CTS determina el dispositivo de comunicación que está dispuesto a recibir datos.

Pin 6 (DSR) : Data Set Ready (from communication equipment). El dispositivo de comunicación indica que tiene datos disponibles para ser emitidos.

Pin 7 : Masa de señal. Es usual en dispositivos electrónicos llevar por separado la masa de señal y la masa de protección (toma protectora de tierra) para reducir las influencias perturbadoras de la red.

Pin 8 (DCD) : Data Carrier Detect (from communication equipment). El dispositivo de comunicación indica que está transmitiendo.

Pin 15 (TxC) : Transmitter Clock (reloj de emisión).

Pin 17 (RxC) : Receiver Clock (reloj de recepción).

Pin 20 (DTR) : Data Terminal Ready (to communication equipment). Se comunica al dispositivo de comunicación la disponibilidad de recepción.

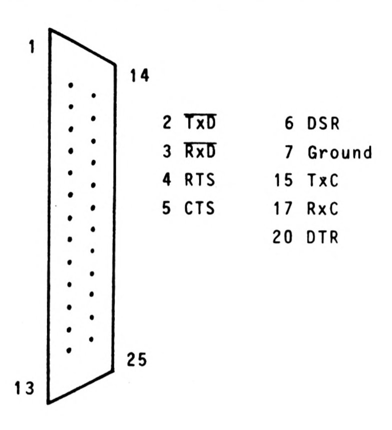


Fig. 6.6 Ocupación de conectores del RS232C (Conectores miniatura D).

Como podrá apreciarse, el estándar posee varias posibilidades de comunicación. El estándar RS232C es de origen americano. Este ha sido definido por la EIA (Electronics Industry Association) y de una forma ligeramente modificada (CCITT-V.24) ha sido aceptado en Europa como norma estándar.

Con el RS232C pueden realizarse funciones simples, semiduplex y duplex completas (o bien duplex cortas). (Simples: únicamente para emitir o recibir; semiduplex: para emitir y recibir en secuencias temporales; duplex: para emitir y recibir al mismo tiempo.)

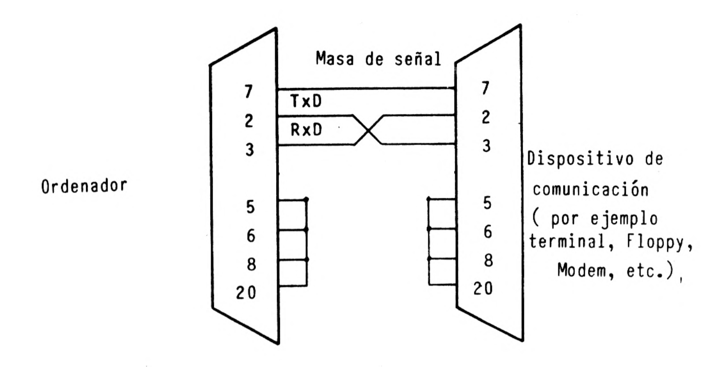


Fig. 6.7 Bucle de retorno automático.

Esta configuración representa un caso especial importante, ya que con un mínimo de líneas permite un funcionamiento duplex completo. Mediante bucles cortos de las líneas de control se simula un "O.K." a las unidades. Como es de suponer esta conexión puede únicamente aplicarse, cuando el dispositivo de comunicación es lo suficientemente rápido para poder leer los datos emitidos por el ordenador. Si el dispositivo de comunicación es una impresora, el ordenador no podrá emitir más caracteres por segundo que los que esté en condiciones de recibir la impresora en el mismo espacio de tiempo (por ejemplo 150

C/s; ésto corresponderia al utilizar un bit de inicio y dos bits de paro para una velocidad máxima de transmisión de 150 x (8 + 1 + 2) = 1650 Bd, es decir para una velocidad máxima de transmisión estándar de 1200 Bd.).

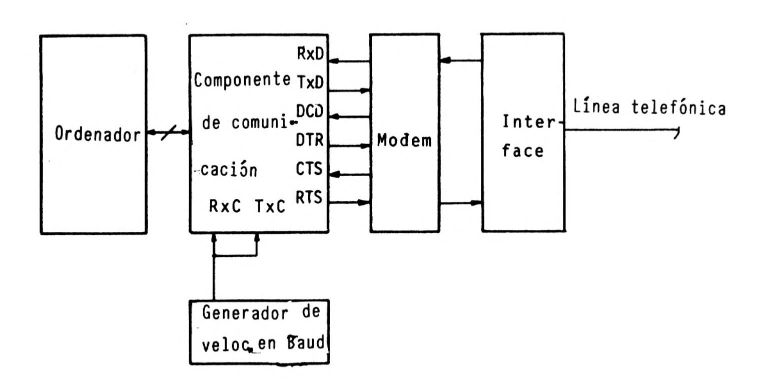


Fig. 6.8 Transmisión de datos asíncronos a través de la línea telefónica.

Las conexiones 4, 5, 6, 8 y 20 pueden utilizarse para el control del modem. Al utilizar el estándar RS232C podrán transmitirse datos hasta una distancia de 15 m directamente sobre líneas. Cuando se trata de medios de transmisión más largas (por ejemplo 10 Km), se utilizarán los llamados modems (Modulator/Demodulator), los

cuales convierten la información digital a las frecuencias correspondientes. A cada estado lógico (0 o bien 1) se le asignará por ejemplo una frecuencia para la modulación de frecuencia. Estas frecuencias son algo distintas al emitir que al recibir, con el fin de evitar acoplamientos; de este modo cuatro frecuencias distintas posibilitarán la transferencia de datos en la línea telefónica. Mediante la conversión de los impulsos de la tensión contínua a corriente alterna de frecuencia alterna, la red de transmisión no será sobrecargada por los elevados saltos de tensión de las señales del RS232C (ondas armónicas superiores) y se aumentará enormemente la seguridad de transmisión, ya que los impulsos de interferencia que hayan podido filtrarse no podrán conllevar a una interpretación errónea por parte del receptor.

La interface de la línea telefónica puede, o bien consistir de un circuito eléctrico directo de adaptación o estar realizado en forma de un acoplador acústico. La solución mediante un acoplador acústico tiene la ventaja de poseer una flexibilidad local, como por ejemplo se exige en los ordenadores "Handheld"; por el otro lado una conexión fija con la línea telefónica tiene la ventaja de poseer una seguridad de transmisión más elevada.

RS 422/RS 423

En el campo industrial las distancias de transmisión entre una micro-unidad de cálculo periférica (casi de proceso) y un ordenador central son de varios centenares

de metros. La aplicación de modems sería en estos casos poco operativa. Por este motivo se utilizan los estándar RS 422 ó RS 423. Contrariamente al RS 232 C no trabajan con tensiones absolutas sinó con tensiones diferenciales:

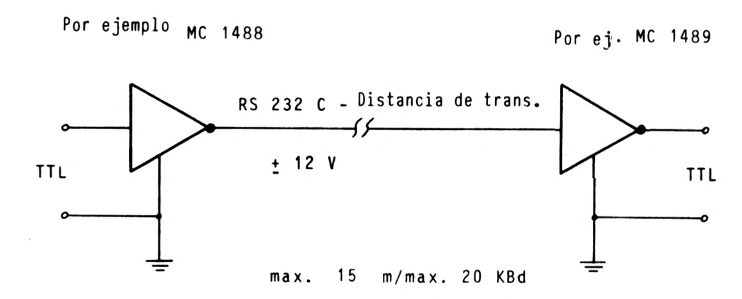


Fig. 6.9 RS 232 C - Unión directa de líneas.

Las tensiones se refieren a una masa común.

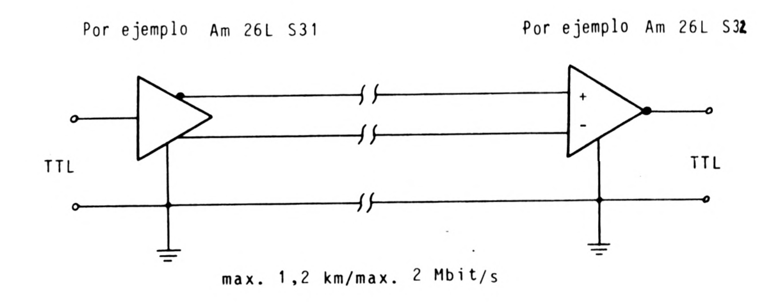


Fig. 6.10 RS 422 - Unión directa de líneas.

En la recepción común sólo se detecta la diferencia de tensión. La radiación de interferencias actúa al mismo tiempo sobre ambas líneas de transmisión, no teniendo efecto alguno. El aumento que de ello resulta en la seguridad de transmisión explica la elevada velocidad de emisión de hasta 2 MBits/s. La tensión mínima de recepción será de 200 mV diferenciales y la corriente máxima de cortocircuito de 100 mA.

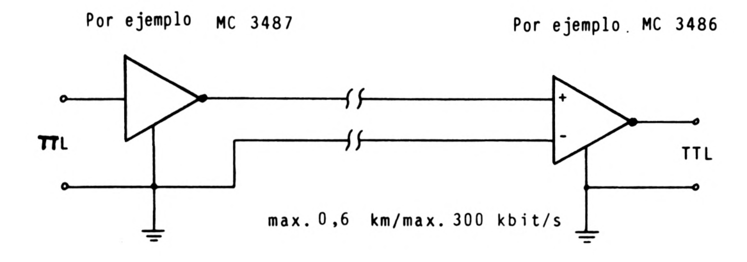


Fig. 6.11 RS 423 - Unión directa de líneas.

El RS 423 trabaja contrariamente al RS 422 con toma de tierra simétrica. La tensión mínima de recepción es de 200 mV diferenciales y la corriente máxima de cortocircuito de 100 mA.

Si se desea utilizar cables baratos para la transmisión de datos y sin embargo no distorsionar las líneas vecinas podrá trabajarse también con semiciclos senoidales para reducir la parte de las ondas armónicas superiores.

El receptor formará de los semiciclos senoidales mediante un Schmitt-Trigger los correspondientes impulsos rectangulares.

El formato de datos podrá ser cualquiera en el RS 232 C, RS 422 y RS 423 (ASCII de 7 bits, ASCII de 7 Bits más paridad, código Baudot de 5 bits, etc.).

Resumen del RS 232 C:

El RS 232 C es el estándar más divulgado, asíncrono y en serie de bits para la transmisión de datos. Su equivalente europeo se llama V.24. Se aplica para velocidades de transmisión hasta 20 KBd (= 20 KBits/s) y permite recorridos directos de transmisión de hasta 15 m. Cuando se trata de distancias mayores se aplican entonces los llamados "Modems". A través de los mismos podrá utilizarse asimismo la red pública de teléfonos para la transmisión de datos, abriéndose así una red de transmisiones a nivel mundial. El RS 232 C trabaja con los niveles de tensión +12V y -12V, asignándose el -12V al nivel lógico "1" y el ±12V al nivel lógico "0". Así pues deberán aplicarse transformadores de niveles del TTL (0V/+5V) al RS 232 C (-12V/+12V) y viceversa, para lo que se dispone de componentes integrados.

El estándar utiliza conectores miniatura D de 25 polos y permite un movimiento simple, semiduplex y duplex completo. Normalmente se utiliza tan sólo una parte de las conexiones (por ejemplo 2, 3, 4, 5, 7, 8 y 20 para el funcionamiento de un modem) o se aplica el bucle

automático de retorno. Existen una serie de componentes de integrados de comunicación que apoyan este estándar (Z80-SIO, 82 51, 82 74, etc.). El formato de transmisión consta de un bit de inicio (lógico 0), de varios bits de datos (generalmente 5-8), a elección de un Bit de paridad y de uno y medio o dos bits de paro. En las pausas las líneas están sobre el "nivel de señal" (lógico 1). La codificación de los caracteres a ser transmitidos no se ha determinado.

En el campo industrial también se aplican los estándar RS 422, y RS 423. Ya que trabajan con tensiones diferenciales y frente a influencias eléctricas de interferencias son relativamente insensibles, pueden aplicarse para distancias de transmisión de hasta 1,2 Km sin tener que utilizar modems. Sus posibles velocidads de transmisión son además muy elevadas (300 KBits/s, respectivamente 2MBits/s).

6.5 SDLC

Además de la transmisión de datos asíncrona y en serie, se aplican también en la práctica la transmisión de datos síncrona y en serie. Mientras que la técnica de transmisión asíncrona es aplicable para velocidads de transmisión hasta 19,2 KBaud (19200 Bits por segundo), los tipos de transmisión síncrona son más indicados para velocidads de Baud más elevadas. La razón se encuentra en la desaparición de los bits de inicio, de paro y de

paridad por cada uno de los bytes de datos a ser transmitidos y además en la posibilidad de igualar la frecuencia de cadencia del bloque a ser transmitido a la de la velocidad en Baud. Para una comunicación asíncrona, la frecuencia de reloj debería ser por lo menos 16 veces más elevada que la velocidad de Baud, para que los pequeños desplazamientos de fases y las diferencias de frecuencia entre el emisor y el receptor no sean motivo de errores de transmisión. En el proceso síncrono de datos se requiere únicamente la intercalación, cada 800 bits, de un carácter de sincronización de ocho bits de largo en la corriente de datos, con el fin de poder sincronizar al receptor en la cadencia de emisión. De hecho, existen varios protocolos usuales sobre la transmisión síncrona de datos.

Los más conocidos son el HDLC (High-Level Data Link Control) y el SDLC (Synchronous Data Link Control) de la Firma IBM. El HDLC se diferencia sensiblemente del SDLC:

1) Mientras el SDLC puede direccionar como máximo "únicamente" 256 dispositivos, el número de dispositivos direccionables del HDLC es ilimitado.

2) El número máximo de Control Fields (campos de control) en el HDLC es de dos y en el SDLC de uno.

3) El HDLC Abort Character (carácter de interrupción) consta de siete unos consecutivos, mientras el SDLC Abort Character se compone de ocho unos consecutivos.

A continuación se tratará únicamente con más detalle el SDLC. Este protocolo de transmisión de datos está

ortadodo por el Z80-SIO. Incluso utilizando otros componentes pueden realizarse el SDLC y otros protocolos con el Z80 (ejemplo: 82 73 Programmable HDLC/SDLC Protocol Controller o bien 82 74 Multi-Protocol Serial Controller de la Firma Intel).

Todos los protocolos síncronos de transmisión de datos están estructurados de forma similar. Constan de un margen subdividido en campos.

Caracteres síncronos	Dirección	Información de control	Datos	CRC	Caracteres síncronos

Fig. 6.12 Marco general para la transmisión síncrona.

Según el contenido de la información de control pueden distinguirse distintos tipos de margenes, por ejemplo:

Margen de información: Sirve para la transmisión de datos del emisor al receptor.

Margen de protocolo: Sirve al management para la transmisión en el circuito de transmisión de datos.

El formato del margen utilizado especialmente por el SDLC tiene el siguiente aspecto:

Flag inicial	Campo direccion.	Campo de control	Campo de inform.	Secuencia de comprobación	Flag final del margen
(F)	(A)	(C)	(I)	(FCS)	(F)
0111 1110 (7EH)	Bit 8	Bit 8	Cualquier longitud 0...N Bits	Bit 16	0111 1110 (7EH)

Fig. 6.13 Formato del margen del SDLC

El margen representado anteriormente muestra un margen de informaciones. Como diferenciación puede mencionarse, que el margen del protocolo no tiene ningún campo de información. Un margen de protocolo se compone de un total de 48 bits en el SDLC. Por lo tanto un margen de información siempre se compone de más de 48 bits. Ya que el SDLC es un protocolo orientado en bits, el campo de información no deberá contener necesariamente un múltiplo par de 8 bits, aunque ésto ocurra muy a menudo en la práctica; más bien, se permite cualquier cantidad de bits, como por ejemplo el 13, el 91, etc.

a) <u>Los campos flag:</u>

Todo margen se finaliza por ambos lados mediante un campo de flag. El flag inicial (Opening Flag) define el principio del campo de dirección. El flag final (Closing Flag) indica final de margen, definiendo así los 16 bits precedentes como campo FCS. Los campos flag sirven para la sincronización del emisor y del receptor. Además, realizan la función de caracteres de relleno de tiempo entre márgenes, es decir, que son emitidos cuando no debe transmitirse ningún margen.

El lector crítico se preguntará si el número de posibles caracteres que pueda contener el margen ha sido reducido por el carácter del flag (7EH). En realidad sería así si el SDLC no tuviese aquí un método rebuscado para evitar esta limitación. Este método se llama "Zero Bit Insertion", dicho de otro modo "inserción de cero bit". El emisor añade siempre un cero (fuera de los campos flag) cuando consecutivamente siguen cinco ceros. De este modo es imposible llegar a la combinación de bits 0111 1110 entre los campos de flag.

El receptor se para, después de haber detectado el flag inicial, cada cero del flujo de datos que va llegando, que sigue a los cinco unos consecutivos, reconstruyendo así la información inicial (que también puede contener el byte 7EH). El flag final es, en primer lugar, el próximo flag que posee seis unos consecutivos, cerrando así el margen.

b) El campo de dirección:

Existe una estación primaria (Primary) y hasta 255 estaciones secundarias (Secundaries). La estación primaria domina los procesos de la red e indica a las estaciones secundarias cuándo y qué tareas han de realizar. Cada estación secundaria puede seleccionarse desde la estación primaria a través de la dirección del campo de dirección. A la estación primaria no se le ha asignado ninguna dirección. Para el caso en que la estación primaria deseara emitir al mismo tiempo, se definirá la dirección correspondiente con FFH, es decir, todo con unos. Si una estación secundaria seleccionada manda a la estación primaria un margen (como respuesta), entonces el campo de dirección contiene la dirección de la estación secundaria que está emitiendo; por lo cual la estación primaria sabrá con exactitud qué estación secundaria, está emitiendo en aquel momento.

c) El campo de control:

El código de control no será reconocido por los componentes síncronos de comunicación, sino que se transmitirá al procesador. El SDLC trabaja tanto bajo el funcionamiento semiduplex como en el duplex. Las propiedades del SDLC son, sin embargo, únicamente explotadas bajo el funcionamiento duplex (emitir y recibir al mismo tiempo), dado que una estación SDLC puede confirmar primero márgenes recibidos, mientras al mismo tiempo emite datos. Pueden confirmarse hasta siete márgenes implícitos, antes de que se retorne una

confirmación (Acknowledgement) explícita. Las siete confirmaciones se mandarán dentro de un margen. Otros métodos, como por ejemplo el tan extendido Bi-Sinc, exigen que cada margen sea dado por recibido mediante un margen de confirmación. De este modo se obtiene una ventaja significativa del SDLC, ya que así se ahorra tiempo. El campo de control contiene, entre otros, la información necesaria para la realización de esta técnica.

d) El campo de información:

El campo de información puede contener cualquier tipo de información. Incluso su longuitud puede seleccionarse a voluntad (por bits).

e) El campo de secuencias de verificación del margen (FCS):

El campo FCS sirve para detectar los errores de transmisión. El contenido del campo FCS es una palabra de verificación CRC de 16 bits (Cyclic Redundancy Check). La palabra de verificación CRC se calcula de la siguiente forma:

La palabra de verificación CRC transferida es el resto cumplimentado que se obtiene, cuando el campo A,C e I se divide por el llamado "polinomio del generador" ($X^{16} + X^{12} + X^5 + 1$). El receptor divide A,C,I y el campo FCS

por el polinomio del generador ($X^{16} + X^{12} + X^{5} + 1$). En el caso de quedar un resto, se confirmará un error de transmisión y la estación receptora enviará al emisor el correspondiente mensaje. Mientras la información redundante que contiene el campo FCS no sea suficiente para la corrección automática de errores, deberá emitirse el margen nuevamente. Todos los márgenes recibidos hasta aquí de forma correcta, serán en esta ocasión igualmente confirmados.

Con el polinomio del generador SDLC $X^{16} + X^{12} + X^{5} + 1$ podrá reducirse, por ejemplo, en una longitud de bloque de 260 bits la cuota de error residual sobre el factor 50000.

f) El flag final:

El flag final tiene el mismo contenido que el flag de inicio y señala el final del margen. Los 16 bits precedentes serán de este modo definidos como campo FCS.

Durante la transmisión de un margen puede ocurrir que, por ejemplo, los datos requeridos en la memoria no estén disponibles o bien haya tenido lugar una interrupción de la señal CTS (Clear to send) del modem. En estos casos y antes de finalizar la transferencia del margen será oportuna la interrupción de la transmisión. Esto se realizará mediante la emisión de un carácter de interrupción (Abort character), el cual consta de ocho unos inmediatamente consecutivos. Tan pronto lea el receptor un carácter de interrupción, interrumpirá el ciclo actual de lectura del margen tratándolo de tal manera,

como si no se hubiese iniciado nunca. Un mensaje al emisor sobre un margen erróneo no es necesario en estos casos, en los cuales el emisor detecta un error durante el proceso de emisión y finaliza la transmisión enviando un carácter de interrupción.

Si el emisor ha enviado los datos correctamente, pero en el curso de la transmisión se han infiltrado errores, será entonces deber del receptor de reconocerlos mediante una verificación FCS. Además, el receptor ignorará todos aquellos márgenes que sean demasiado cortos (menos de 32 bits entre los campos flag) o bien aquéllos en los que falten campos (por ejemplo un flag de cierre).

Las argumentaciones dadas hasta el momento sobre el SDLC se referían a redes "Point-to-Point" (punto a punto) y "Multi-Point" (multipunto).

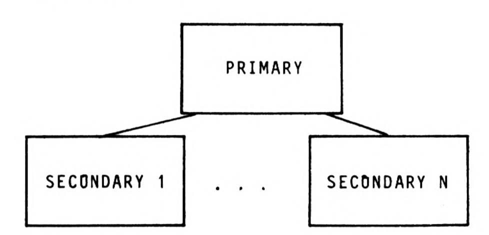


Fig. 6.14 Redes Point-to-Point.

Cada estación secundaria está conectada mediante una línea propia a la estación primaria. Esta red se aplicará cuando las estaciones secundarias sean instaladas en una constelación de espacios dispersos. Además las conexiones son independientes entre sí. Cuando por ejemplo la línea de la primera estación secundaria es defectuosa, ésto no influirá en las demás estaciones secundarias.

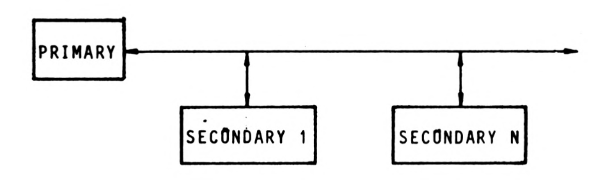


Fig. 6.15 Red Multi-Point

La construcción de una red Multi-Point es de utilidad cuando las estaciones secundarias estén ubicadas en línea. En esta constelación, las estaciones secundarias serán conectadas a un bus de serie común. En el caso de que el bus estuviese defectuoso en alguna parte, se "desconectarán" de la comunicación de datos las estaciones conectadas a continuación. Por otro lado se ahorrarán líneas.

Resumen del SDLC:

Las redes SDLC constan de una estación primaria y hasta 255 estaciones secundarias. Actualmente sólo podrá emitir una estación, pero todas podrán recibir a un mismo tiempo (dirección FFH). La estación primaria decide lo que hará una estación secundaria y cuándo lo realizará. Las informaciones se intercambiarán siempre en forma de margen. Estos márgenes se componen de cinco hasta seis campos. El inicio y final del margen está siempre representado por un flag (para la sincronización del emisor y del receptor). Una estación secundaria únicamente acepta un margen, si éste contiene su dirección en el campo de dirección (o bien si en el mismo se encuentra la dirección general FFH). Si se acepta un margen se verificará a continuación si está libre de errores. Podrán aceptarse hasta siete márgenes sin errores antes de que tenga lugar un mensaje de retorno (el margen contiene las siete confirmaciones). El SDLC es un método de transmisión de datos muy efectivo y adecuado para la red.

6.6 Z80-SIO (Serial Input/Output)

Con este componente se pone a disposición del usuario una herramienta extraordinariamente variada y potente para la transmisión de datos en serie. Además de la comunicación asíncrona este componente soporta los estándars más importantes para la transmisión síncrona para la que están siempre dispuestos dos canales independientes entre si y aptos para el duplex completo. Cuando se trata de una frecuencia de reloj de 4 MHz pueden transmitirse con el SIO hasta 880 K-Bits/s (880.000 bits por segundo). En el funcionamiento asíncrono el SIO ofrece las siguentes caracteristicas de rendimiento:

1.) Los caracteres transmitidos pueden constar de 5, 6, 7 o bien 8 bits (programables).

2.) Hay 1, 1 1/2 o bien 2 bits de paro programables.

3.) Al carácter puede seguirle un bit de paridad que puede controlar paridad par o bien impar. Sin embargo puede omitirse el bit de paridad.

4.) Las marcas se crean automáticamente (emisión) o bien siendo también reconocidas (recepción).

5.) El SIO reconoce asímismo errores de paridad, de desbordamiento y de margen.

6.) Subdivisión del reloj del canal sobre los factores 1, 16, 32 o bien 64.

Otro caso de aplicación realizable mediante el SIO es la comunicación síncrona utilizando el SDLC o bien el protocolo de transmisión HDLC. Como ya sabemos, existen entre el SDLC y el HDLC diferencias mínimas. Para ambos

protocolos el SIO ofrece las siguientes características de rendimiento:

1.) Creación y reconocimiento de los caracteres de interrupción (abort character).

2.) Inserción y eliminación automática de ceros después de (más de) cinco unos consecutivos.

3.) Inserción automática de flags entre transmisiones.

4.) Reconocimiento del campo de dirección.

5.) Protección de las informaciones válidas recibidas ante un desbordamiento.

6.) Generación y verificación de los bytes CRC.

El SDLC y el HDLC son protocolos, como es sabido, orientados en bits. El componente SIO permite, sin embargo, la transmisión síncrona orientada en bytes, es decir, en el Monosync especial, Bisync y External Sync. Los formatos de transmisión de estos protocolos están indicados en la Fig. 6.15. La diferencia más evidente del formato de transmisión SDLC/HDLC es la ausencia de un campo de dirección y de un carácter de sincronización al final de la transmisión (Closing Flag). Las características de rendimiento más importantes del SIO para estos protocolos son las siguientes:

1.) Sincronización de caracteres interna o externa.
2.) Aplicación de uno o dos caracteres de sincronización en registros separados.
3.) Inserción automática de los caracteres de sincronización.
4.) Generación y verificación de los bytes CRC.

En general el SIO ofrece la posibilidad de selección entre el CRC-16 y el polinomio CRC-CCITT.

Ambos canales SIO disponen de líneas de control para el funcionamiento del Modem. De modo similar al Z80-PIO, el Z80-SIO tiene también características de interrupts vectoriales y lógica de prioridades de Daisy Chain.

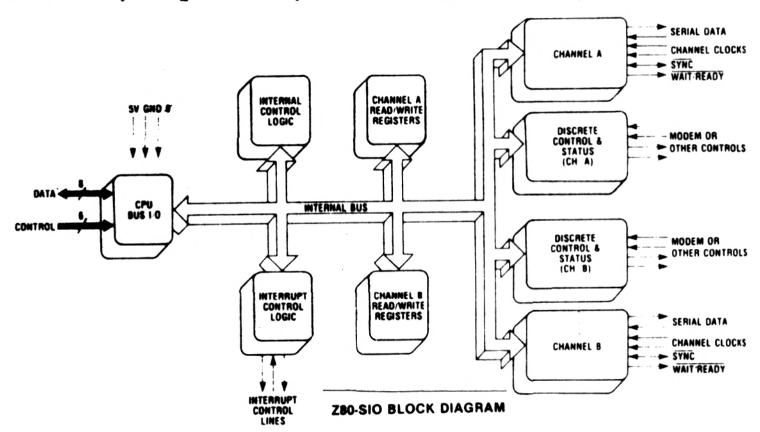


Fig. 6.15 Esquema de bloques del Z80-SIO.

Debido a la limitación de 40 conexiones (pins) tuvieron que aceptarse para el canal B ciertas limitaciones. Zilog ofrece tres versiones SIO distintas (SIO/0, SIO/1 y SIO/2), con el fin de poder resolver los problemas particulares de cada usuario.

SIO/0: TxCB y RxCB están conectados entre si.
SIO/1: DTRB falta.
SIO/2: SYNCB falta.

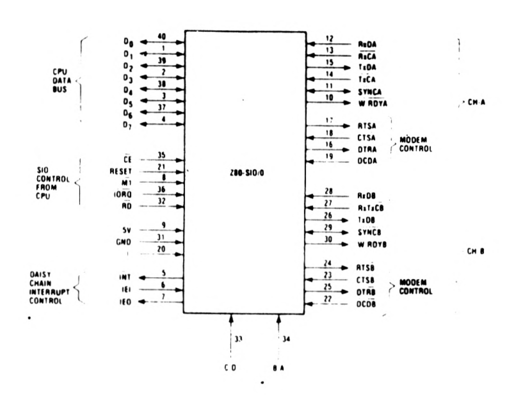


Fig. 6.16 Ocupación de pins del Z80-SIO/0

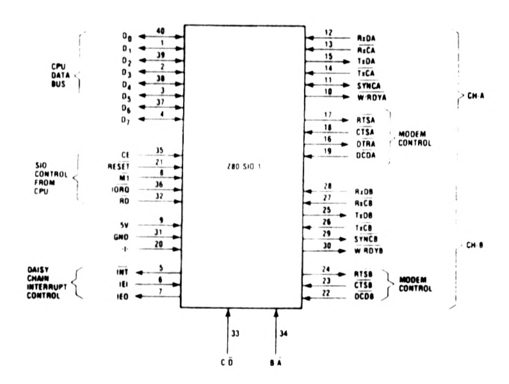


Fig. 6.17 Ocupación de pins del Z80-SIO/1

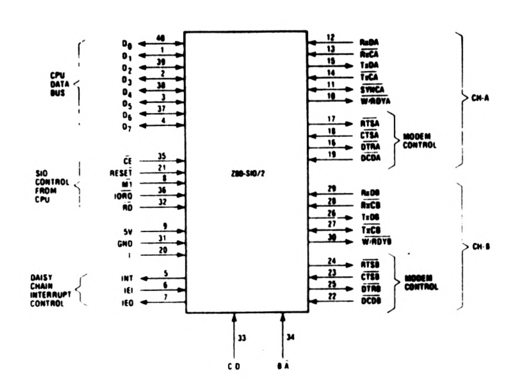


Fig. <u>6.18</u> Ocupación de pins del Z80-SIO/2

<u>Descripción de pins del Z80-SIO:</u>

D0 hasta D7 : Estas líneas se conectan con el bus de datos del sistema Z80 y posibilitan así la transferencia de palabras de datos y de control entre la CPU-Z80 y el SIO-Z80.

B/A El estado lógico de esta entrada de control define si el canal B (lógico 1) o el canal A (lógico 0) es seleccionado por la CPU. Muy a menudo se conecta la entrada con A0 del bus de direcciones de la CPU.

C/D Si esta entrada es lógico 1, se seleccio-

nan en caso de una transferencia entre CPU y Z80-SIO los registros de códigos de control del canal seleccionado a través de B/A.

Si la entrada es lógico 0, se activarán los registros de datos del canal correspondiente. La entrada define el tipo de transferencia entre la CPU y el Z80-SIO: Transferencia de instrucciones o de datos. Generalmente ésta se conecta con A1 del bus de direcciones de la CPU.

CE (Chip Enable) Si esta entrada es lógico 0, el Z80-SIO podrá comunicarse con la CPU. La entrada controla el circuito de decodificación (desde D7 hasta D2) o bien, en pequeños sistemas, una de las líneas de direcciones de la CPU A7 hasta A2.

O (Reloj del sistema) La entrada se conecta sencillamente con el reloj del sistema de la CPU.

M1 (Machine Cycle 1) La entrada se conecta con M1 de la CPU del Z80 y se interpreta por el mismo junto con el RD y el IORQ (reconocimiento de un ciclo de extracción o bien confirmación de un interrupt).

IORQ (Input/Output Request) Esta entrada se conectará con el IORQ del Z80-µP e indicará que la CPU desea comunicarse con un componente de entrada/salida. Junto con el M1 sirve además para reconocer la confirmación de un interrupt de la CPU.

RD

(Read) Esta entrada se conectará con el RD del Z80-μP e indicará que la CPU está realizando una operación de lectura.

RESET

Con esta entrada el Z80-SIO puede inicializarsecon un estado de salida determinado:

1.) Las dos salidas de emisión TxDA y TxDB obtienen niveles de marca (lógico 1).

2.) Todas las salidas de control del modem serán lógico 1.

3.) Todos los interrupts están desactivados

El Z80-SIO deberá inicializarse de nuevo después de cada RESET.

IEI

(Interrupt Enable In) Esta entrada se necesita cuando más de un componente puede provocar interrupts dentro de una cadena de prioridad Daisy-Chain. Unicamente cuando esta entrada es lógico 1, podrá el Z80-SIO crear interrupts (ésta se habrá liberado para interrupts).

IEO

(Interrupt Enable Out) Esta salida se utilizará, tal como la IEI, en relación con la lógica de prioridad Daisy-Chain. Se conecta con IEI de la siguiente cadena, bloqueando así todos los componentes posteriores (con prioridad inferior), mientras sea lógico 0. La IEO será entonces lógico 1, cuando IEI del

SIO es lógico 1 y también si la CPU no envía ninguna rutina de interrupción al SIO.

INT (Interrupt Request) Esta salida de Open-Drain será conectada con la entrada INT del Z80-µP así como con las salidas INT de los componentes restantes.

W/RDYA,W/RDYB (Wait/Ready A, Wait/Ready B) Estas dos entradas podrán realizar dos funciones distintas independientemente de la programación:

1.) Función Wait: La CPU será sincronizada con las palabras de datos del SIO. (Salida Open-Drain.)

2.) Función Ready: Las salidas indicarán un DMAC, por si el SIO puede transmitir o recibir datos.

CTSA, CTSB (Clear to Send) También estas dos entradas son programables en cuanto a su función:

1.) Función Auto-Enable: Las entradas sirven para la liberación automática del emisor, la cual tendrá lugar cuando se dispone de un nivel lógico 0.

2.) Función General-Purpose: Las entradas son generalmente utilizables.

DCDA, DCDB (Data Carrier Detect) Para estas entradas sirve lo indicado para las entradas CTS,

pero con la diferencia de que son programables para la liberación automática de receptores.

RxDA, RxDB — (Receive Data) Entradas de datos en serie.

TxDA, TxDB — (Transmit Data) Salida de datos en serie.

RxCA, RxCB — (Receiver Checks) Entradas de reloj para la velocidad en Baud del receptor. La frecuencia de reloj es programable a la velocidad de Baud de 1, 16, 32 o bien 64 veces (de tipo funcional asíncrono).

TxCA, TxCB — (Transmitter Checks) Entradas de reloj para la velocidad de Baud del emisor. Los factores del divisor deberán ser iguales para el emisor y el receptor. Cuando se trata de instalaciones de frecuencias distintas podrán, sin embargo, elegirse en las entradas de cadencia componentes de emisión y de recepción.

RTSA, RTSB — (Request to Send) En la modalidad síncrona estas salidas seguirán estrictamente el estado del bit RTS. En la modalidad asíncrona dichas salidas podrán inicializarse en cualquier momento a lógico 0 (RTS=lógico 1); después del lógico 1 será posible cuando el RTS se ha desactivado (RTS=lógico 0) y si además el buffer de emisión esté vacío. También podrán ser utilizadas como salidas comunes.

DTRA, DTRB — (Data Terminal Ready) Estas salidas seguirán estrictamente el estado del bit

DTR. También podrán ser utilizadas como salidas comunes.

SYNC A, SYNC B (Synchronisation) Estos pins pueden ser tanto entradas como salidas. Según la modalidad, las funciones serán las siguientes:

1.) Modalidad de recepción asíncrona:
Los pins sirven de entradas e influyen en el bit Sync/Hunt del registro de lectura RR0.

2.) Modalidad síncrona externa:
Los pins sirven asímismo de entradas y deberán inicializarse a través de la lógica externa a lógico 0, cuando después de la llegada del último bit de sincronización hayan pasado 2 ciclos completos de reloj de recepción. La entrada deberá activarse nuevamente a lógico 1, cuando tenga que transmitirse un nuevo margen o bien cuando se pierda la sincronización y la CPU informe debidamente a la lógica externa. La sincronización de caracteres se realizará en esta modalidad externamente.

3.) Modalidad síncrona interna:
(Monosync, Bisync) Los pins trabajan como salidas y serán siempre lógico 1, cuando una muestra de caracteres de sincronización sea detectada.

Función Z80-SIO:

La función fundamental del Z80-SIO es la de un conversor/controlador paralelo/serie, a serie/paralelo. Con el fin de poder dirigir las múltiples posibilidades de este componente, dispone éste de un gran número de registros de escritura/lectura. El canal A posee siete registros de escritura (WR 0, WR 1, WR 3, WR 4, WR 5, WR 6, WR 7) y dos registros de lectura (RR 0, RR 1). El canal B dispone de otros dos registros, o sea el WR 2 y el RR 1 (en total 8 registros de escritura y dos registros de lectura). El Z80-SIO ofrece tanto la posibilidad del polling como también la de interrupts, donde el canal A será internamente de mayor prioridad que el canal B. Los registros de lectura darán información sobre el estado del SIO. Para grabar los registros de escritura se requieren (con la excepción del WR 0) dos bytes. El primero sirve como indicador de aquel registro de escritura en el que deberá grabar el segundo byte, escribiéndose siempre en el WR 0. De importancia es el hecho que los registros del vector de interrupción (WR 2 y RR 2) únicamente están previstos para el canal B. Ya que el vector de interrupción será modificado por el SIO (bits D3, D2, D1), tanto el canal B como también el canal A serán capaces de provocar interrupts (a cada canal se le han asignado cuatro tipos de interrupción).

La Fig. 6.19 indica una representación resumida de las funciones bit del registro de escritura.

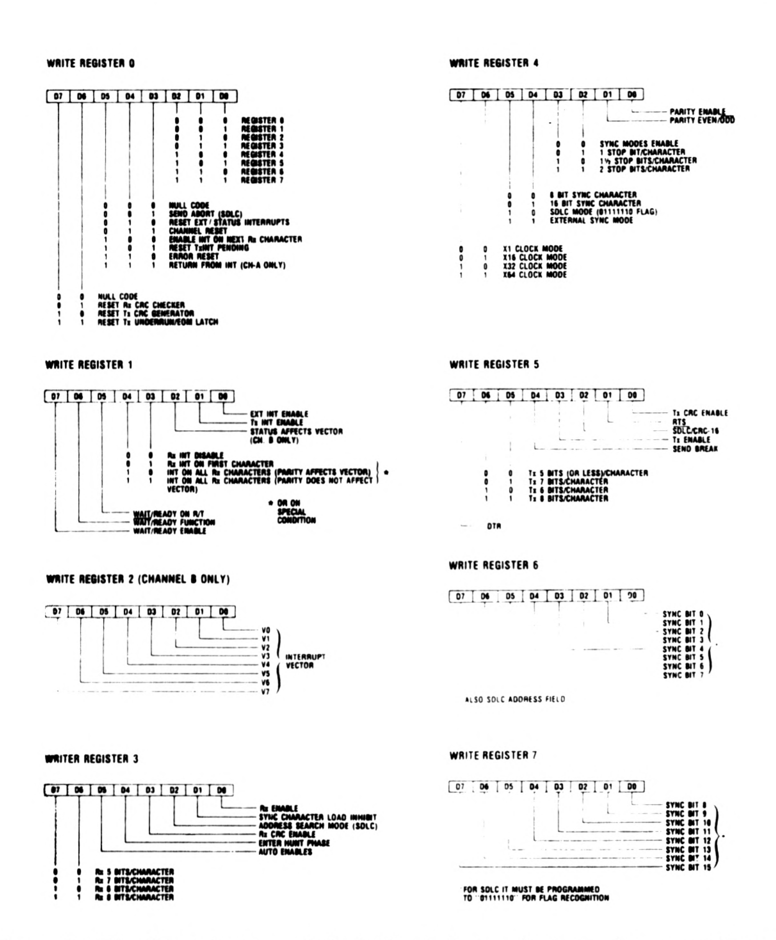


Fig. 6.19 Funciones del bit del registro de escritura

La función de los distintos registros de escritura (bits) es la siguiente:

Registros de escritura 0

Los bits D0 hasta D2 se utilizan como apuntador al siguiente registro que deba grabarse o leerse. Después de cada reset y también después de cada refencia a un registro (menos el WR 0), el apuntador señala automáticamente al WR 0.

Los bits de instrucción D3 hasta D5 definen las siete instrucciones básicas del SIO.

La instrucción RESET TxINT PENDING hace que el emisor no genere más interrupts en el caso de que el buffer del emisor esté vacío (en la transferencia con la modalidad Interrupt Enable). Cuando el buffer de emisión se carga y se vacía nuevamente, podrá volverse a generar un interrupt.

La instrucción ERROR RESET desactiva los bits D5 y D4 del RR 1 (errores de paridad y de desbordamiento).

La instrucción RETURN FROM INT tiene el mismo efecto que la instrucción RETI-CPU y da preferencia al canal B ante el canal A.

La instrucción NULL CODE no tiene efecto alguno.

Registro de escritura 1

El bit D0 (Ext. Int. Enable) posibilita la generación de bnterrupts a través de las entradas DCD, CTS y SYNC mediante el reconocimiento o finalización de caracteres de Break/Abort o mediante el inicio de una transmisión de caracteres CRC o de sincronización, cuando se activa

Transmit Underrun/EOM-Latch (RR 0, bit D6).

Cuando el bit D2 (Status Affects Vector) se desactiva, el SIO trabaja únicamente con un vector de interrupción, o sea con aquél que se ha almacenado en el WR 2. Sin embargo si el bit D2 ha quedado activado, el SIO modifica el vector en las posiciones V3, V2 y V1 dependiendo del tipo de estado que lo origina o bien de los acontecimientos.

V3	V2	V1	Acontecimiento/estado	Canal
0	0	0	Buffer de emisión vacío	B
0	0	1	Modificación externa del estado	B
0	1	0	Recibido carácter	B
0	1	1	Error de paridad, de recepción, de desbordamiento y de margen, así como final del margen (SDLC)	B
1	0	0	Buffer de emisión vacío	A
1	0	1	Modificación externa del estado	A
1	1	0	Recibido carácter	A
1	1	1	Error de paridad, de recepción, de desbordamiento y de margen, así como final del margen (SDLC)	A

Una modificación externa del estado se obtiene al cambiar el estado de una de las líneas de estado CTS, DCD o bien SYNC.

Con el bit D7 (Wait/Ready Enable) se conectará/desconectará la función WAIT/READY.

El bit D6 (Wait/Ready Enable) selecciona entre la función de Wait y de Ready.

El bit D5 (Wait/Ready on R/T) determina si la función Wait/Ready se refiere al estado del buffer emisor (lleno/vacío) o bien al estado del buffer receptor (lleno/vacío).

D7	D6	D5	Acontecimiento/estado	Estado de WAIT/READY
0	1	X	./.	1
0	0	X	./.	Alto ohmiaje
1	1	0	Buffer emisor lleno	1
1	1	0	Buffer emisor vacío	0
1	0	0	Buffer emisor lleno y selección del Port de datos del SIO	0
1	0	0	Buffer emisor vacío	Alto ohmiaje
1	1	1	Buffer receptor vacío	1
1	1	1	Buffer receptor lleno	0
1	0	1	Buffer receptor vacío y selección del Port de datos del SIO	0
1	0	1	Buffer receptor lleno	Alto ohmiaje

Registro de grabación 3

El bit DO (Rx Enable) da paso al receptor y debería activarse una vez se hayan programado los restantes parámetros.

Si el bit D1 (Sync Character Load Inhibit) es lógico 1, los caracteres de sincronización del inicio no se cargarán en el buffer receptor.

Si el bit D2 (Adress Search Mode (SDLC)) ha sido activado y se ha inicializado el canal para el SDLC, únicamente se recibirán márgenes si contienen ya sea la dirección global del FFH o bien la dirección que consta en el WR 6.

El bit D4 (Enter Hunt Phase) se activa automáticamente después de cada reset, pero también podrá activarse si en la modalidad síncrona se ha perdido la sincronización o en la modalidad SDLC no se requiere una transmisión. Si el Bit D4 ha sido activado, el SIO esperará la llegada de un carácter de sincronización (también se activa el Bit D4 en el RR 0).

Si el D5 (Auto Enables) ha sido activado, servirán los CTS de la liberación del receptor y los DCD de la liberación del emisor. De lo contrario serán entradas normales, cuyos estados aparecen en los bits D4 y D4 del RR 0.

Registro de grabación 5

El bit D1 (RTS) determina en la modalidad síncrona el estado de la línea RTS. En la modalidad asíncrona la línea RTS vendrá después del lógico 1, una vez se haya emitido el carácter por completo y el buffer emisor esté vacío.

El bit D2 (SDLC/CRC-16) selecciona el polinomio generador:

Bit D2 = high: CRC-16 ($X^{16} + X^{15} + X^{2} + 1$)
Bit D2 = low : SDLC ($X^{16} + X^{12} + X^{5} + 1$)

Los bits D5 y D6 determinan el número de bits de que se

componen los caracteres a ser emitidos (en la modalidad asíncrona).

El bit D7 (DTR) sirve como bit de control para la salida DTR.

La CPU debería emitir los caracteres al SIO en el formato siguiente:

D7	D6	D5	D4	D3	D2	D1	D0	Bits/caracteres
D	D	D	D	D	D	D	D	8
X	D	D	D	D	D	D	D	7
X	X	D	D	D	D	D	D	6
0	0	0	D	D	D	D	D	5
1	0	0	0	D	D	D	D	4
1	1	0	0	0	D	D	D	3
1	1	1	0	0	0	D	D	2
1	1	1	1	0	0	0	D	1

Registro de grabación 6

La función de este registro depende de la modalidad:

Modalidad	Contenido del registro
Monosync	Carácter de sincronización de emisión de 8 bits
External Sync	Carácter de sincronización de emisión de 8 bits
Bisync	Primer carácter de sincronización de 8 bits
SDLC	Campo de dirección de 8 bits (del secundario)

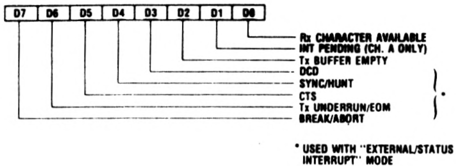
Registro de grabación 7

La función de este registro también depende de la modalidad:

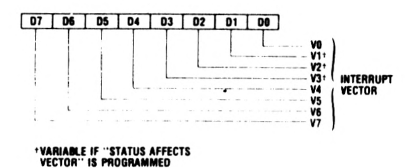
Modalidad	Contenido del registro
Monosync	Carácter de sinc. de recepción de 8 bits
External Sync.	(No utilizado)
Bisync	2º carácter de sincronización de 8 bits
SDLC	Carácter del flag (0111 1110) 7EH

Aparte de los registros de grabación están también los registros de lectura del Z80-SIO accesibles para el programador. Estos contienen informaciones sobre errores que hayan tenido lugar, señales estándar de comunicación del interface y sobre el vector actual del interrupt (y sobre modificaciones). La Fig. 6.20 contiene una representación resumida de las funciones Bit del registro de lectura.

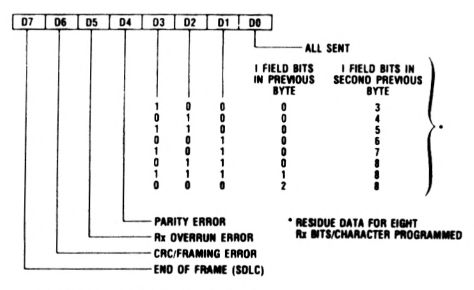
READ REGISTER 0



READ REGISTER 2



READ REGISTER 1 †



† USED WITH SPECIAL RECEIVE CONDITION MODE

Fig. 6.20 Funciones bit del registro de lectura

El proceso de lectura de los registros de lectura funciona de manera análoga a la grabación en los registros de grabación.

La función de los diferentes registros de lectura (bits) es la siguiente:

<u>Registros de lectura 0</u>

El bit D1 (Int. Pending (Ch.A Only)) es únicamente efectivo en el canal A; en el canal B es siempre Low. Se activa siempre en el canal A, cuando en el Z80-SIO se haya cumplido cualquier condición de interrupt. Se aplica en sistemas que no puedan procesar interrupts vectorizados, porque entonces deja de trabajar mientras la rutina de servicio verifica individualmente todos los bits de los registros RR 0.

El bit D3 (DCD) indica el estado de la entrada DCD en el momento de la última modificación de uno de los 5 bits externos del estado (DCD, CTS, Sync/Hunt, Break/Abort o Transmit Underrun/EOM).

El bit D4 (Sync/Hunt) realiza distintas funciones que dependen de la modalidad.

Modalidad asíncrona: El bit D4 funciona como el bit D3 (DCD), pero se refiere a la entrada SYNC.

Modalidad externa síncrona: Cuando el bit Enter Hunt Phase ha sido activado, la lógica externa deberá mantener la entrada SYNC en lógico 1 hasta que se haya realizado la sincronización. Si se ha

desactivado, la función corresponderá a la modalidad asíncrona.

Modalidades Monosync, Bisync: El bit D4 se activa en primer lugar mediante el bit Enter Hunt Phase (el bit Enter Hunt Phase será activado por la CPU al final de un margen y al perder la sincronización). El bit D4 se desactiva tan pronto se haya sincronizado el SIO. El paso high-low del D4 ocasiona una interrupción externa de estado que debe ser desactivado por la CPU mediante la emisión de la instrucción "Reset External/-Status Interrupt" (WT 0). La salida SYNC será siempre lógico 0, si se recibe un carácter de sincronización.

Modalidad SDLC: El bit D4 se activa mediante el bit Enter Hunt Phase, pero también cuando se haya bloqueado el receptor. Se desactiva tan pronto el SIO haya detectado el flag inicial del último margen, no necesitando después, contrariamente al Monosync y Bisync, ser activado explícitamente.

Bit D5 (CTS): Funciona análogamente al bit DCD.

Bit D7 (Break/Abort): Tiene únicamente aplicación en las modalidades asíncronas de recepción y en el SDLC.

Registros de lectura 1

Bit D0 (All Sent): Se refiere únicamente a la modalidad asíncrona. No se desencadenan interrupts.

Bits D1, D2 y D3: Estos bits tienen únicamente importancia si se activa el bit de End of Frame. Indican la longitud del campo I siempre y cuando no sea un múltiplo par de la longitud del carácter.

Bit D5 (Receive Overrun Error): Indica que se han recibido más de tres caracteres sin haber sido recogidos por la CPU (el Z80-SIO contiene en la parte receptora una memoria FIFO con una profundidad de tres bytes).

Registros de lectura 2

Dependiendo del bit 2/WR 1, este registro contiene el vector grabado en el WR 2 o bien el vector modificado por el SIO de las posiciones V3, V2 y V1.

Como pequeña ayuda la tabla 6.21 indica los bits que deberán tener un contenido determinado en las distintas modalidades, de modo que en la programación SIO no deberán ser motivo de duda:

Modalidad	Registro	D7	D6	D5	D4	D3	D2	D1	DØ
SDLC	WR 3							Ø	
	WR 4	Ø	Ø	1	Ø	Ø	Ø	Ø	Ø
	WR 5				Ø		Ø		
MONO-, BI- and EXT.-SYNC.	WR 3						Ø		
	WR 4	Ø	Ø			Ø	Ø		
	WR 5						Ø		
Asyn-chron	WR 3				Ø	Ø	Ø	Ø	
	WR 4			Ø	Ø				
	WR 5						Ø		Ø

Tabla 6.21 Estados bit de registros predeterminados

Los registros de grabación WR 6 y WR 7 no han sido necesarios para la comunicación asíncrona.

6.7 Ejemplo con el Z80-SIO:

Para la demostración de cómo debe aplicarse un Z80-SIO para la comunicación en serie, deberá inicializarse en el siguiente ejemplo el canal A para la transmisión síncrona de datos en el formato Bisync y el canal B para la transmisión asíncrona. El SIO podrá estar en el campo 6CH hasta 6FH y ambos canales deberán trabajar con Interrupts modificados.

Las direcciones del port del SIO son:

6CH – datos del canal A
6DH – datos del canal B
6EH – control del canal A
6FH – control del canal B

El esquema de bloque queda representado en la Fig. 6.22.

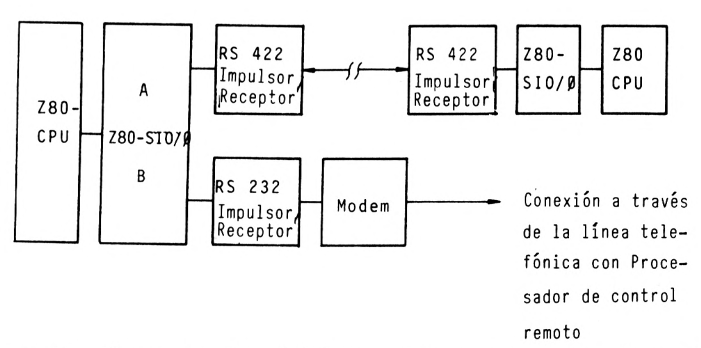


Fig. 6.22 Esquema de bloque para la comunicación en serie del procesador.

El canal A trabaja en la modalidad Bisync y el canal B en la modalidad asíncrona.

Rutina de inicialización del SIO:

```
; Desactivación (reset) de ambos canales (A y B)

         DI              ; Bloquear interrupts del Z80-CPU
         LD A, 58H
         OUT (6EH),A     ; WR 0, reset del canal, reset del
                           RxCRC
         LD A, 18H
         OUT (6FH),A     ; WR 0, reset del canal B


; Recibir el canal de inicialización A-Bisync

         LD A, 02H       ;
         OUT (6EH),A     ; WR 0, indicador en WR 2
         LD A, 80H
         OUT (6FH),A     ; Vector del interrupt 80H, can. B
         LD A, 04H
         OUT (6EH),A     ; WR 0, indicador en WR 4
         LD A, 10H
         OUT (6EH),A     ;  x  1,  Bisync,  ningún  bit  de
                           paridad
         LD A, 15H
         OUT (6EH),A     ; Indicador en  WR  5,  Reset Ext.
                           Int.
         LD A, 64H
         OUT (6EH),A     ; CRC-16
         LD A, 03H
         OUT (6EH),A     ; Indicador en WR 3
         LD A, FAH
         OUT (6EH),A     ; 8 bits/carácter, Sync Char. Load
                           contenido
```

```
; Receive CRC Enable Enter Hunt Made, Auto Enables

         LD A,06H
         OUT (6EH),A     ; Indicador en WR 6
         LD A,05H
         OUT (6EH),A     ; Primer carác. de sincronización
                           05H
         LD A,07H
         OUT (6EH),A     ; Indicador en WR 7
         LD A,0AH
         OUT (6EH),A     ; 2º  carácter  de sincronización
                           0AH
         LD A,15H
         OUT (6EH),A     ; Indicador  en  WR 1, reset ext.
                           interrupt
         LD A,1CH
         OUT (6EH),A     ; Int.  en  cada  carácter  (sin
                           prioridad)
; el vector es modificado a través del estado
; canal de inicialización modalidad asíncrona B para
; bits  de paro  de 1 1/2, 8 bits Rx, 8 bits Tx, paridad
; par, x 32
; Interrupts del receptor y emisor, estado modificado
; Vector
;        LD A, 14H
         OUT (6FH),A     ; Indicador en  WR  4,  Res. Ext.
                           Int.
         LD A, 8BM
         OUT (6FH),A     ; x  32, bits  de paro  de 1 1/2,
                           Paridad par
         LD A, 03H
         OUT (6FH),A     ; Indicador en WR 3
         LD A, E1H
         OUT (6FH),A     ; 8 bits de Rx/carácter
; Auto Enables
         LD A,05H
         OUT (6FH),A     ; Indicador en WR 5
         LD A,EAH
         OUT (6FH),A     ; 8 bits de Tx/carácter, liberar
```

```
; emisor,
         RTS, DTR
         LD A,11H
         OUT (6FH),A     ; Indicador  en WR  1, reset Ext.
                           Int.
         LD A,17H
         OUT (6FH),A     ; Canal B finalizado

; inicializado, tanto el emisor como el receptor
; pueden generar interrupts

         LD A,13H
         OUT (6EH),A     ; Indicador en WR 3, canal A
         LD A,FBH
         OUT (6EH),A     ; Canal A finalizado

; para la modalidad Bisync/recibido inicializado y
; puede crear los correspondientes interrupts

         EI              ; Liberar interrupts del Z80-CPU
```

En nuestra rutina de inicialización hemos cargado el vector del interrupt 80H = (1000 0000B). Para que en el caso de interrupts éstos puedan ser debidamente manipulados, deberán constar en las siguientes direcciones las rutinas de manejo de interrupts (el registro I del Z80-CPU contiene, por ejemplo, 40H):

Dirección:	Rutina de servicio:	Canal:
4000H	Buffer emisor vacío	B
4002H	Modif. ext. del estado	B
4004H	Recibido caracteres	B
4006H	Estado esp. de recepción	B
4008H	Buffer emisor vacío	A
400AH	Modif. ext. del estado	A
400CH	Recibido caracteres	A
400EH	Estado esp. de recepción	A

El canal B se encuentra en la modalidad asíncrona y podrá trabajar sin ningún otro tipo de inicialización en modalidad duplex completo. El canal A se encuentra en la modalidad síncrona (Bisync) y por este motivo únicamente podrá aplicarse a modalidad semiduplex. Si se desea emitir a través del canal A, éste deberá ser previamente inicializado. Mediante la aplicación de instrucciones de carga en bloque podrán fácilmente manipularse las rutinas necesarias de inicialización.

Tal como podrá apreciarse, el Z80-SIO es un componente bastante complejo con muchas posibilidades de aplicación. Con todas estas representaciones se han tratado los puntos más importantes para su aplicación práctica.

7. Componentes de contador/timer del Z80-CTC

7.1 Generalidades:

El Z80-CTC es un componente especial con el que se pueden contar impulsos o bien generar intérvalos de tiempo. El componente está equipado con la lógica del Interrupt de prioridad Daisy-Chain, típica del sistema Z80, que ya hemos conocido en el Z80-SIO. El componente ofrece cuatro canales independientes entre si. Los componentes de contador/timer proporcionan las referencias exactas de tiempo requeridas para muchas aplicaciones, que no pudieron ser garantizadas, respectivamente puestas a disposición por el Z80-CPU, en cuanto al software.

El Z80-CTC está equipado con la cadencia 0 del sistema, la cual será dividida en la modalidad de timer por un subdivisor interno por el factor 16 o bien por el factor 256. Esta cadencia (subdividida) disminuirá un contador interno descendente de 8 bits que podrá generar un Interrupt si su contenido es cero. Además se emitirá un impulso (Zero Count/Time Out). En la modalidad de timer la función temporal podrá ser impulsada por flancos señalizadores tanto positivos como negativos (Clock/-Trigger). En la modalidad de contador todo flanco activo señalizador en la entrada CLK/TRG hace decrecer el contador descendente de 8 bits. Cuando el contador ha disminuido hasta cero, el canal podrá generar un Interrupt y emitir un impulso positivo (ZC/TO).

7.2 Z80-CTC:

Arquitectura del Z80-CTC:

La Fig. 7.1 representa el diagrama en componente del CTC. El Z80-CTC consta de un bus-interfase del Z80-CPU, de una lógica interna de control, de una lógica de control de interrupt y de cuatro canales independientes entre si.

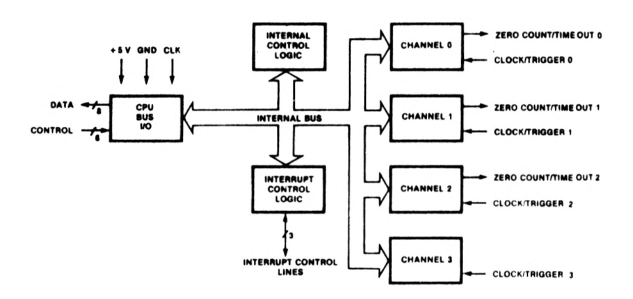


Fig. 7.1 Diagrama de bloque del Z80-CTC.

La estructura de cada uno de los canales está indicada en la Fig. 7.2. Podremos distinguir un registro de control del canal, un registro de constantes de tiempo de 8 bits, un subdivisor de 8 bits y un contador descendente de 8 bits. El canal 3 no tiene ninguna salida Zero Count/Time, debido a la limitación de 28 pins. Por lo demás, los cuatro canales son funcionalmente idénticos.

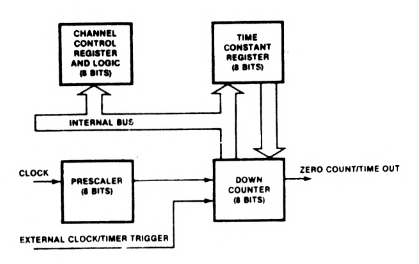


Fig. 7.2 Diagrama de bloque del canal Z80-CTC

Descripción de los pins del Z80-CTC

El Z80-CTC está dispuesto en una caja Dual-In-Line de 28 polos. La Fig. 7.3 muestra la configuración de los pins del CTC y la Fig. 7.4 presenta la configuración de la caja del CTC.

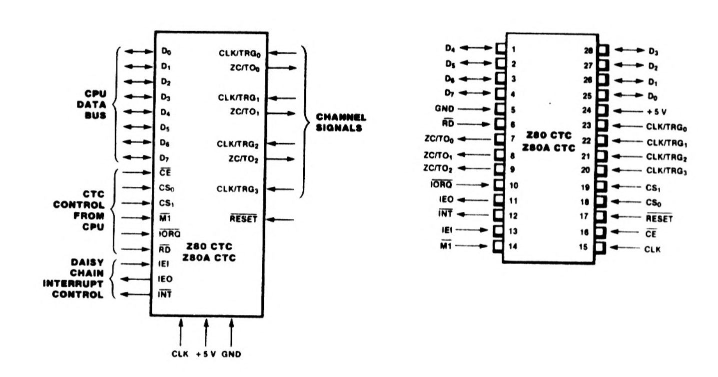


Fig. 7.3
Config. de los pins del CTC

Fig. 7.4
Config. de la caja del CTC

D7 - D0 Bus de datos del sistema

CS1-CS0 (Channel Select) Entradas de selección de canal. Generalmente se conectan estas entradas con las líneas de dirección A1 y A0. La tabla de verdad es:

CS1	CS0	Canal
0	0	0
0	1	1
1	0	2
1	1	3

CE (Chip Enable) Esta entrada es generalmente alimentada por un decodificador de direccio-

nes, que está conectado a los ocho bits inferiores del bus de direcciones.

CLK — (Clock, 0) Reloj del sistema.

M1,RD,IORQ — Estas líneas se conectan con las del mismo tipo del Z80-CPU. Su función ha sido ya descrita (CPU, SIO, PIO).

IEI,IEO,INT — Estas líneas posibilitan el encadenamiento a una Daisy-Chain.

RESET — Desactivación del CTC en estado definido de salida. Todos los contadores serán parados, todos los bits de liberación de interrupt serán puestos a cero y todas las salidas serán llevadas al estado de alto ohmiaje.

CLK/TRG 3
CLK/TRG 0 — (External Clock/Timer Trigger) En la modalidad de tiempo un borde de impulso activo señalizador inicia la función del contador en dichas entradas. En la modalidad de contador, todo borde de impulso activo señalizador decremente en esta entrada el contador de decrementos de 8 bits. El estado activo del borde de impulso señalizador es programable a positivo (ascendente) o bien a negativo (descendente).

ZC/TO 2
ZT/TO 1 — (Zero Count/Timeout) Siempre que el contador de decrementos de 8 bits decrementa a cero, se produce un impulso en esta salida. (El impulso es activo lógico 1.)

Los impulsos se emitirán tanto en la modalidad de contador como de timer. La duración de los impulsos corresponde a la duración del período del ciclo del sistema.

La modalidad de contador del Z80-CTC (Counter Mode)

En esta modalidad decrementa en 1 el contador de decrementos interno de 8 bits con cada borde de impulso activo de la entrada del CLK/TLG del canal correspondiente. Después de la inicialización del canal para la modalidad de contador se cargará el registro de las constantes de tiempo. El contenido del registro de las constantes de tiempo formará el valor inicial del contador de decrementos. Una vez hayan llegado suficientes borde de impulsos activos y el contador de decrementos a cero, el canal de la salida ZC/TOO emitirá un impulso positivo generando un interrupt, siempre y cuando el canal haya sido liberado para interrupts. Aquí deberá mencionarse que los cuatro canales tienen prioridad interna (el canal 0 tiene la máxima prioridad, el canal 3 la mínima). Además, el contador de decrementos será nuevamente cargado de forma automática con el contenido del registro de las constantes de tiempo de 8 bits. Las frecuencias de llegada de los bordes de impulsos activos están limitadas a máximo la mitad de las frecuencias de reloj del sistema.

La modalidad de timer del Z80-CTC

En esta modalidad el reloj del sistema pasará a través de un subdivisor, antes de llegar al contador de decre-

mentos de 8 bits. Hay dos factores de división que son programables: el 16 y el 256. Los registros de las constantes de tiempo y los contadores de decrementos funcionan como en la modalidad de contador. La función de timer podrá iniciarse, según la programación, de dos maneras:

En primer lugar podrá hacerlo automáticamente depués de la emisión del registro de las constantes de tiempo y en segundo lugar mediante un borde de impulso activo de señalización en la entrada CLK/TRG. Cada vez que se haya decrementado el contador de decrementos a cero, se cargará automáticamente el contenido del registro de las constantes de tiempo en el contador de decrementos, generando al mismo tiempo un interrupt del canal (siempre y cuando haya sido liberado) y dando un impulso positivo en la salida ZC/TO. La frecuencia de los impulsos emitidos se calcula de la frecuencia del reloj del sistema según sigue:

$$F_{Imp} = \frac{F\emptyset}{K.Z}$$

; F_{Imp} = Frecuencia de impulsos

$F\emptyset$ = Frecuencia del reloj del sistema

K = Factor del subdivisor (16 o bien 256)

Z = Constantes de tiempo (1-256)

De forma inversa, se calcula la duración del período de impulsos de la duración del periodo del ciclo del sistema según la relación:

$T\ Imp = TO\ .\ K\ .\ Z$; siendo

Componentes de contador/timer del Z80-CTC

T Imp = Duración del período de impulsos
TO = Duración del período del ciclo del sistema
K = Factor del subdivisor (16 o bien 256)
Z = Constantes de tiempo (1-256)

Tengan presente que el registro de las constantes de tiempo deberá estar cargado con 00H para conseguir una constante de tiempo de 256 (ciclos de reloj).

Interrupts del Z80-CTC

El Z80-CTC envía en el caso de un interrupt (modalidad 2 del Z80-CPU, capítulo 4) un vector de interrupción de 8 bits de ancho al bus de datos. Los bits D2 y D1 del vector serán modificados automáticamente por el CTC, dependiendo del canal que provoca la interrupción, y con la máxima prioridad, del siguiente modo:

D7	D6	D5	D4	D3	D2	D1	D0	Canal
X	X	X	X	X	0	0	0	0
X	X	X	X	X	0	1	0	1
X	X	X	X	X	1	0	0	2
X	X	X	X	X	1	1	0	3

El bit D0 es siempre lógico 0.

Programación del Z80-CTC

El programador podrá programar tres registros por canal:

Registro de códigos de control
Registro de las constantes de tiempo
Registro de vector de interrupt

El registro de códigos de control y el registro del vector de Interrupt podrán programarse directamente (el bit D0 sirve para la selección). El registro de las constantes de tiempo es únicamente accesible, si ello ha sido previamente acordado mediante el código de control (bit D2). Cada canal utilizado deberá ser inicializado mediante un código de control y una constante de tiempo, antes de que pueda iniciar su actividad. Si el Z80-CTC debe generar Interrupts, bastará con cargar una sola vez el registro del vector del Interrupt de cualquiera de los cuatro canales. El CTC modificará automáticamente el vector para los cuatro canales.

El registro de códigos de control

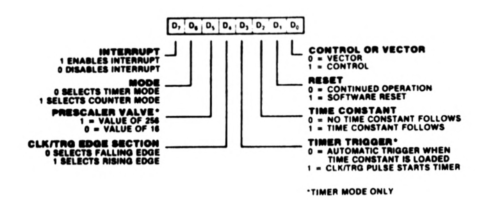


Fig. 7.5 Registro de códigos de control de un canal

Los diversos bits de un código de control, grabado en un registro de códigos de control, tienen el siguiente significado:

D0 = 1 : Con ello podrá identificarse el dato como código de control.

D1 = 1 : Software-Reset. El canal se desactiva y se finaliza la operación de timer/contador. El contenido del registro de código de control se mantendrá. Si D1 y D2 son lógico 1, el canal continuará su actividad tan pronto como el registro de las constantes de tiempo haya sido cargado.

D1 = 0 : El canal proseguirá con su trabajo.

D2 = 1 : El canal cargará el próximo dato que le haya sido grabado en su registro de constantes de tiempo. Una nueva constante de tiempo programada durante el funcionamiento será cargada después del próximo paso de ceros del contador de decrementos.

D2 = 0 : El canal no espera ningún dato para el registro de las constantes de tiempo.

D3 = 1 : La operación de timer se iniciará mediante un borde de impulso activo de señalización en la entrada CLK/TRG. Este bit es únicamente eficaz en la modalidad de timer.

D3 = 0 : La operación de timer empezará automáticamente después de que el registro de las constantes de tiempo haya sido cargado.

D4 = 1 : Modalidad de timer: El borde de impulsos positivo inicia la operación de timer.
Modalidad de contador: El borde de impulsos positivo disminuye el contador de decrementos.

D4 = 0 : Modalidad de timer: El borde de impulsos negativo inicia la operación de timer.
Modalidad de contador: El borde de impulsos negativo disminuye el contador de decrementos.

D5 = 1 : Factor del subdivisor =256 (únicamente eficaz en la modalidad timer)

D5 = 0 : Factor del subdivisor =16 (únicamente eficaz en la modalidad timer).

D6 = 1 : El canal trabaja en la modalidad de contador.

D6 = 0 : El canal trabaja en la modalidad de timer.

D7 = 1 : El canal genera cada vez un interrupt (específico de canal), cuando el contador de decrementos de 8 bits llega a cero.

D7 = 0 : El canal está bloqueado para interrupts.

<u>El registro de las constantes de timer</u>

Si el bit D2 del código de control precedente es lógico 1, el canal cargará el dato siguiente en el registro de las constantes de tiempo. La constante de tiempo podrá estar en el campo de 00H hasta FFH. El 00H corresponde

a una constante de tiempo de 256, el 01H a una constante de tiempo de 1, etc. El contador de decrementos se cargará en primer lugar con las constantes de tiempo y después, dependiendo de la modalidad,se decrementará. Si el contador de decrementos llega a cero, recibirá automáticamente el valor del (posiblemente reprogramado) registro de las constantes de tiempo.

El registro de vector de interrupt

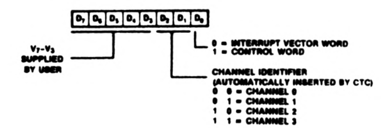


Fig. 7.6 Registro de vector de interrupt

Si el bit D0 del dato grabado para el canal es lógico cero, el canal leerá los bits D7 hasta D3 cargándolos en su registro de vector de interrupt. Ya que estos cinco bits sirven para los cuatro canales, dicho dato deberá ser grabado una sola vez en uno de los cuatro canales. El D2 y el D1 serán automáticamente asignados por el CTC al canal que provoca la interrupción. El D0 es siempre lógico 0.

El contador de decrementos de 8 bits

Cada canal podrá ser leído en cualquier momento. Esto es, por ejemplo, muy importante en sistemas con Polling.

7.2 Ejemplo con el Z80-CTC

Con el fin de poder observar la inicialización de un Z80-CTC, partiremos de la constelación indicada en la Fig. 7.7.

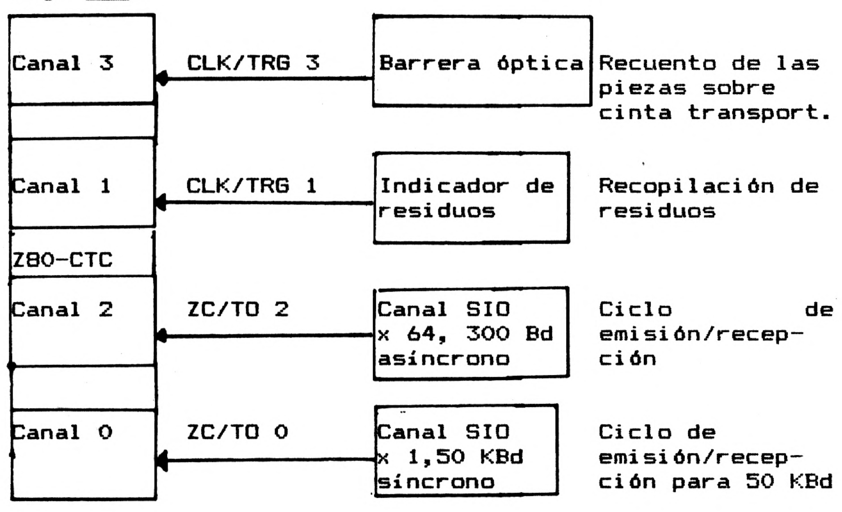


Fig. 7.7 Ejemplo para la aplicación del Z80-CTC

La frecuencia del sistema es de 4 MHz.

Los canales CTC 0 y 2 proveen los dos canales de un SIO con las señales de ciclo requeridas por el emisor/-receptor, es decir 19,2 KHz para el canal SIO 2 y 50 KHz para el canal SIO B. El canal A deberá ser alimentado con 19,2 KHz, ya que queremos partir de un factor interno de división del SIO de 64 (300.64 = 19200). De aquí resultarán los siguientes parámetros para la inicialización de los canales CTC 0 y 2:

Canal 0 : Factor del subdivisor = 16
Constante de tiempo = 13

La velocidad de Baud resultante se calcula con 4.10^6 : 16 : 13 : 64 = 300, 48 (Bd).

Canal 2 : Factor del subdivisor = 16
Constante de tiempo = 5

La velocidad de Baud resultante será de 4.10^6 : 16 : 5 = 50.10^3 = 50 (KBd).

Los canales CTC 1 y 3 trabajan en la modalidad de contador y se han liberado para los interrupts. Los canales 0 y 2 deberán ser consecuentemente bloqueados para los interrupts.

Los distintos canales CTC podrán estar en las siguientes direcciones:

Canal 0 : 50 H
Canal 1 : 51 H
Canal 2 : 52 H
Canal 3 : 53 H

Tanto después de 200 piezas contadas como también después de 10 residuos deberá crearse un interrupt. Como borde de impulso activo deberá definirse el borde de impulso negativo.

El programa de inicialización para el CTC podrá ser:

```
        DI              ; Bloquear la CPU para los interrupts
; Inicialización
  Canal 0
        LD A,07H        ; Código de control para el canal 0
        OUT (50H),A     ; Salida hacia el canal 0
        LD A,05H        ; Constantes de tiempo para canal 0
        OUT (50H),A     ; Salida hacia el canal 0
; Inicialización
  Canal 1
        LD A, A7H       ; Código de control para el canal 1
        OUT (51H),A     ; Salida hacia el canal 1
        LD A, 0AH       ; Constantes de tiempo para canal 1
        OUT (51H),A     ; Salida hacia el canal 1
```

```
; Inicialización
  Canal 2
       LD A, 07H      ; Código de control para el canal 2
       OUT (52H),A    ; Salida hacia el canal 2
       LD A, 13H      ; Constantes de tiempo para canal 2
       OUT (52H),A    ; Salida hacia el canal 2
; Inicialización
  Canal 3
       LD A, A7H      ; Código de control para el canal 3
       OUT (53H),A    ; Salida hacia el canal 3
       LD A, C8H      ; Constantes de tiempo para canal 3
       OUT (53H),A    ; Salida hacia el canal 3
; Carga del vector
  de interrupt 90H
       LD A, 90H      ; Vector de interrupt 90H
       OUT (53H),A    ; Salida hacia el canal 3
       EI             ; Liberar la CPU para los interrupts
```

8. Instrucciones completas del Z80

En las próximas descripciones se utilizan las siguientes abreviaciones y símbolos:

	Estado lógico del flag	
Nombre del flag	0	1
Carry (acarreo) "C"	NC (no carry)	C (carry)
Sign (signo) "S"	P (plus)	M (minus)
Zero (cero) "Z"	NZ (non zero)	Z (zero)
Parity (paridad)/ "P/V" Overflow (desbordamiento)	PO (parity odd)	PE (parity even)

Estados del flag después de la ejecución de la instrucción:

Símbolo	Significado
X	El flag ha sido modificado según la operación correspondiente.
1	El flag ha sido activado (lógico 1).
0	El flag ha sido desactivado (lógico 0).
?	El estado del flag no está determinado.

El registro de estado:

7	6	5	4	3	2	1	0
S	Z	-	H	-	P/V	N	C

Los flags 5 y 3 no están definidos. El flag 1 no tiene significado para el programador.

Los contenidos del registro y de la memoria se indican en los siguientes ejemplos de forma hexadecimal.

ADC HL,ss — Suma HL y el registro par ss con acarreo (carry)

Función: HL <—— HL + ss + C

Flags:

S	Z	H	P/V	N	C
X	X	X	X	O	X

Descripción: Al contenido del par de registros HL se sumarán el operando de 16 bits ss y el contenido del carryflag C. El resultado será almacenado en el par de registros HL.

Ejecución:

Tiempo de ejecución	: 3,75 µs
Frecuencia de reloj	: 4,00 MHz
Estados de reloj	: 15
Ciclos de máquina	: 4

Direccionamiento: Implícito

Formato:

Byte 1) 1 1 1 0 1 1 0 1 (EDH)

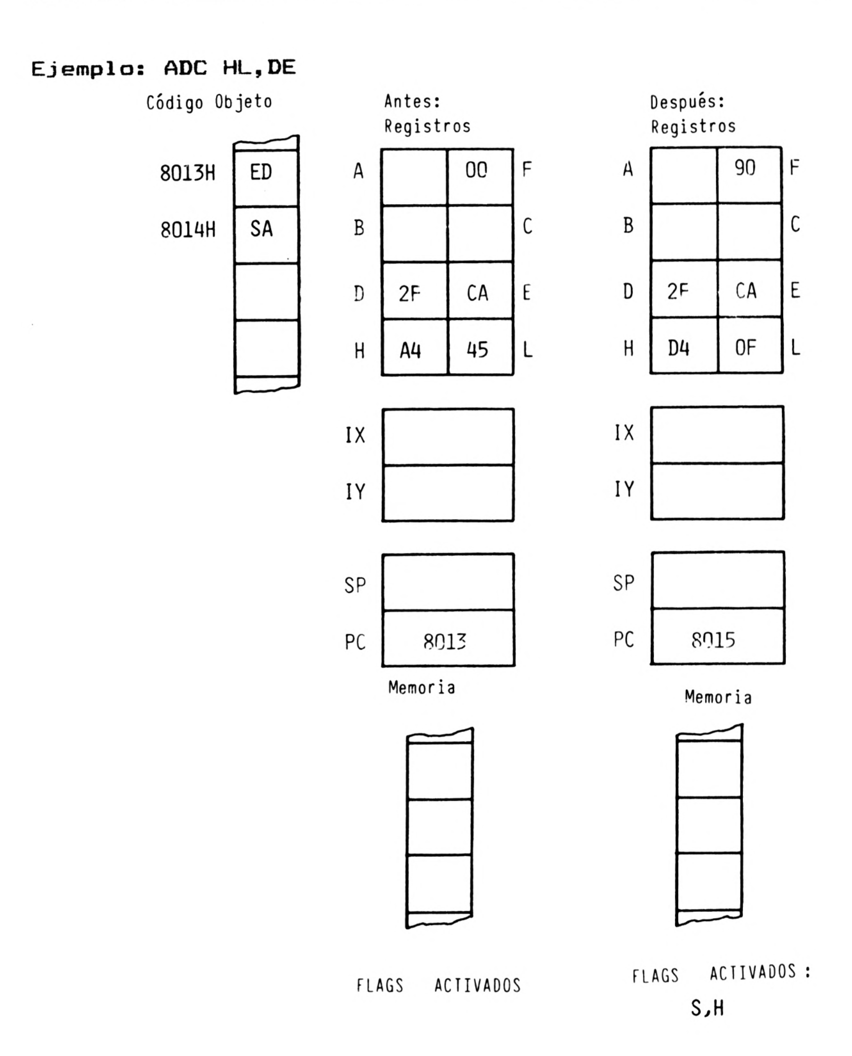
Byte 2) 0 1 s s 1 0 1 0

ss puede ser:

BC - 00 HL - 10

DE - 01 SP - 11

Ejemplo: ADC HL,DE



ADC A,s — Suma el acumulador A y los operandos s con acarreo (Carry).

Función: A <—— A + S + C

Flags:

S	Z	-	H	-	P/V	N	C
X	X		X		X	0	X

Descripción: Al contenido del acumulador A se sumarán el operando s y el contenido del carryflag C. El resultado será almacenado en el acumulador.

Ejecución:

s	µs (4,00 MHz)	Estados T	Ciclos M
r	1,00	4	1
n	1,75	7	2
(HL)	1,75	7	2
(IX + d)	4,75	19	5
(IY + d)	4,75	19	5

Direccionamiento: r: implícito/ n: inmediato/ (HL): indirecto/ (IX+d),(IY+d): indexado

Formato: s: puede ser r,n,(HL),(IX + d), o bien (IY + d)

r:		1	0	1	0	0	---r---			
n:	Byte 1)	1	1	1	0	0	1	1	0	E6H
	Byte 2)	-----------n----------								datos inmedia.
(HL) :		1	0	1	0	0	1	1	0	96H
(IX+d) :	Byte 1)	1	1	0	1	1	1	0	1	DDH
	Byte 2)	1	0	1	0	0	1	1	0	96H
	Byte 3)	-----------d----------								Offset
(IY+d) :	Byte 1)	1	1	1	1	1	1	0	1	FDH
	Byte 2)	1	0	1	0	0	1	1	0	96H
	Byte 3)	-----------d----------								Offset

r puede ser:

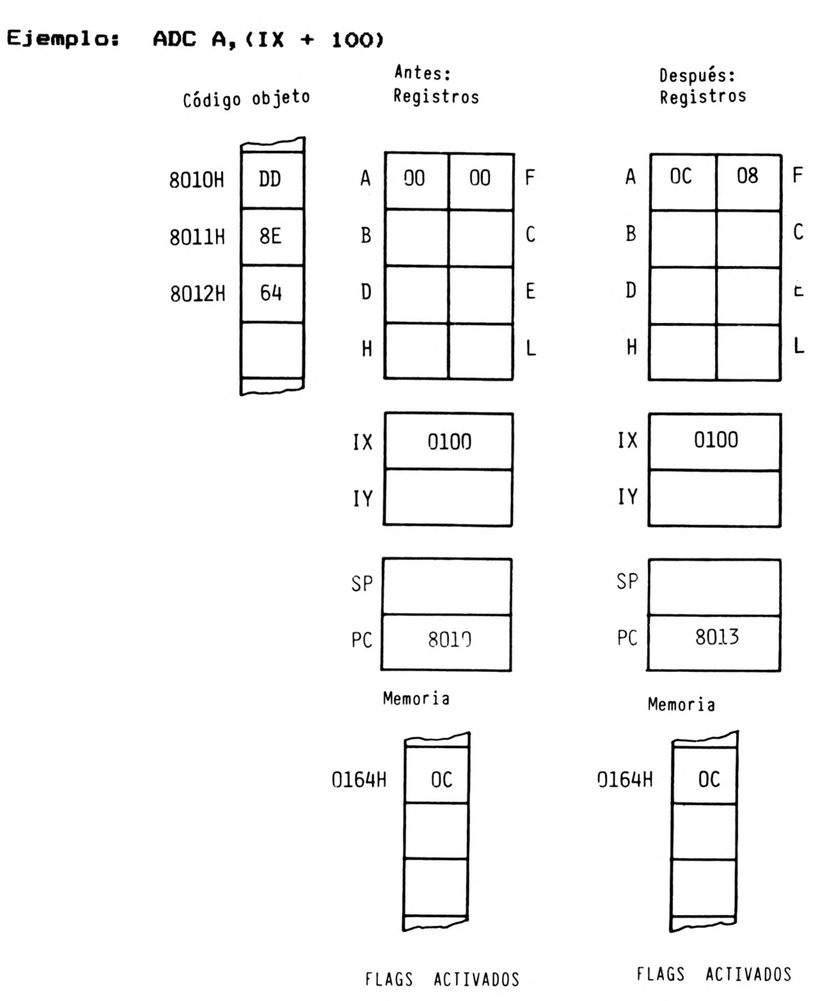
A - 111 E - 011

B - 000 H - 100

C - 011 L - 101

D - 010

Ejemplo: ADC A,(IX + 100)



ADD A,(HL) — **Suma el acumulador con la posición de memoria (HL) direccionada indirectamente a través del HL.**

Función: A <—— A + (HL)

Flags:

S	Z	-	H	-	P/V	N	C
X	X		X		X	0	X

Descripción: Al contenido del acumulador A se sumará el operando (HL) indirectamente direccionado a través del par de registros HL. El resultado será almacenado en el acumulador A.

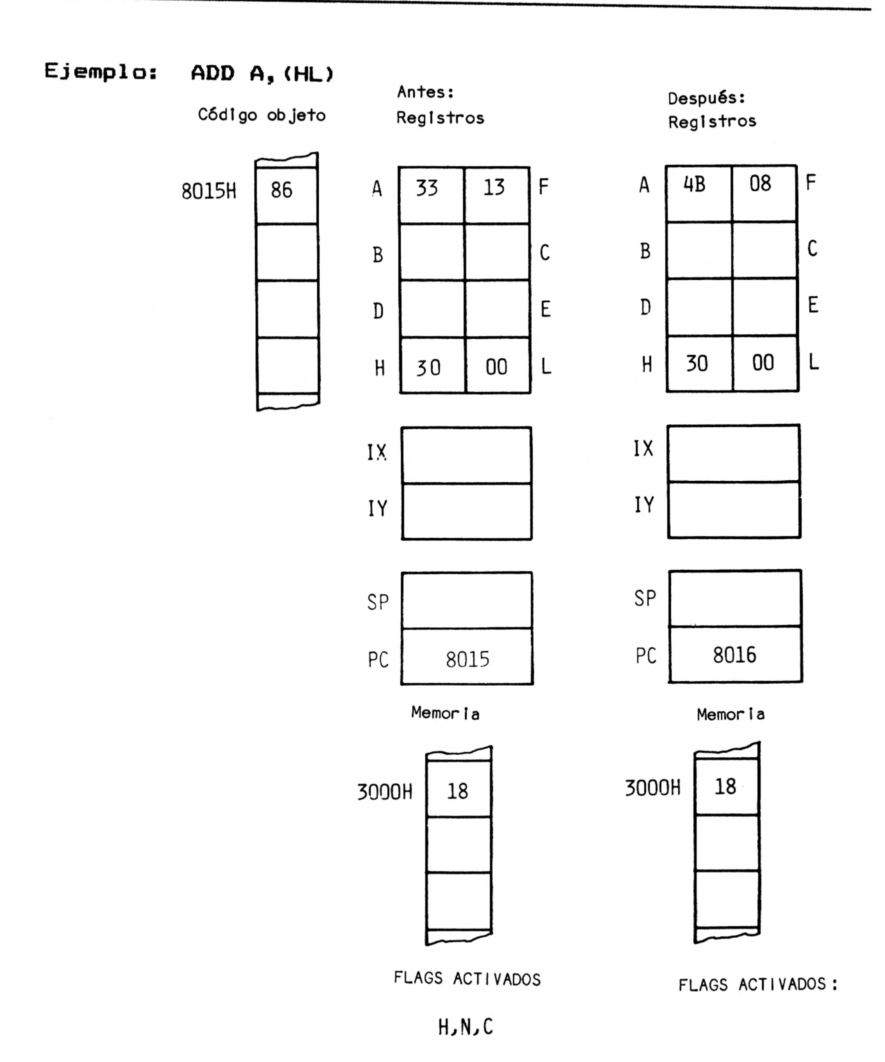
Ejecución:

Tiempo de ejecución	: 1,75 µs
Frecuencia de reloj	: 4,00 MHz
Estados de reloj	: 7
Ciclos de máquina	: 2

Direccionamiento: Indirecto

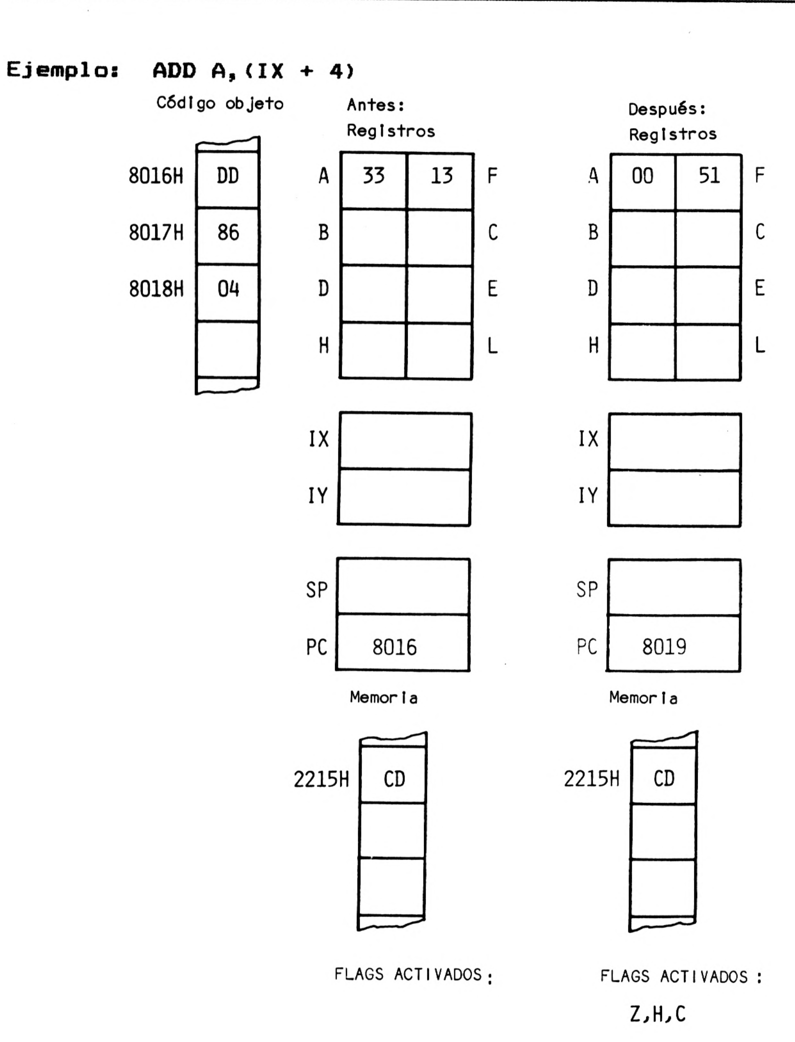
Formato: 1 0 0 0 0 1 1 0 86H

Ejemplo: ADD A,(HL)



ADD A,(IX + d)	Suma el acumulador con la posición de memoria (IX + d) direccionada indexadamente a través de IX + d.
Función:	A <—— A + (IX + d)
Flags:	S Z - H - P/V N C X X X X 0 X
Descripción:	Al contenido del acumulador A se le sumará el operando direccionado indexadamente a través del par de registros IX además del desplazamiento d. El resultado se almacenará en el acumulador.
Ejecución:	Tiempo de ejecución : 4,75 μs Frecuencia de reloj : 4,00 MHz Estados de reloj : 19 Ciclos de máquina : 5
Direccionamiento:	Indexado
Formato:	Byte 1) 1 1 0 1 1 1 0 1 DDH Byte 2) 1 0 0 0 0 1 1 0 86H Byte 3) ---------d------------ Offset

Ejemplo: ADD A,(IX + 4)



<u>ADD A,(IY + d)</u> — Suma el acumulador con la posición de memoria direccionada indexadamente (IY + d).

Función: A <—— A + (IY + d)

Flags:

S	Z	-	H	-	P/V	N	C
X	X		X		X	0	X

Descripción: Al contenido del acumulador A se le sumará el operando direccionado indexadamente a través del par de registros IY además del campo de de desplazamiento d. El resultado será almacenado en el acumulador.

Ejecución:

Tiempo de ejecución	: 4,75 µs
Frecuencia de reloj	: 4,00 MHz
Estados de reloj	: 19
Ciclos de máquina	: 4

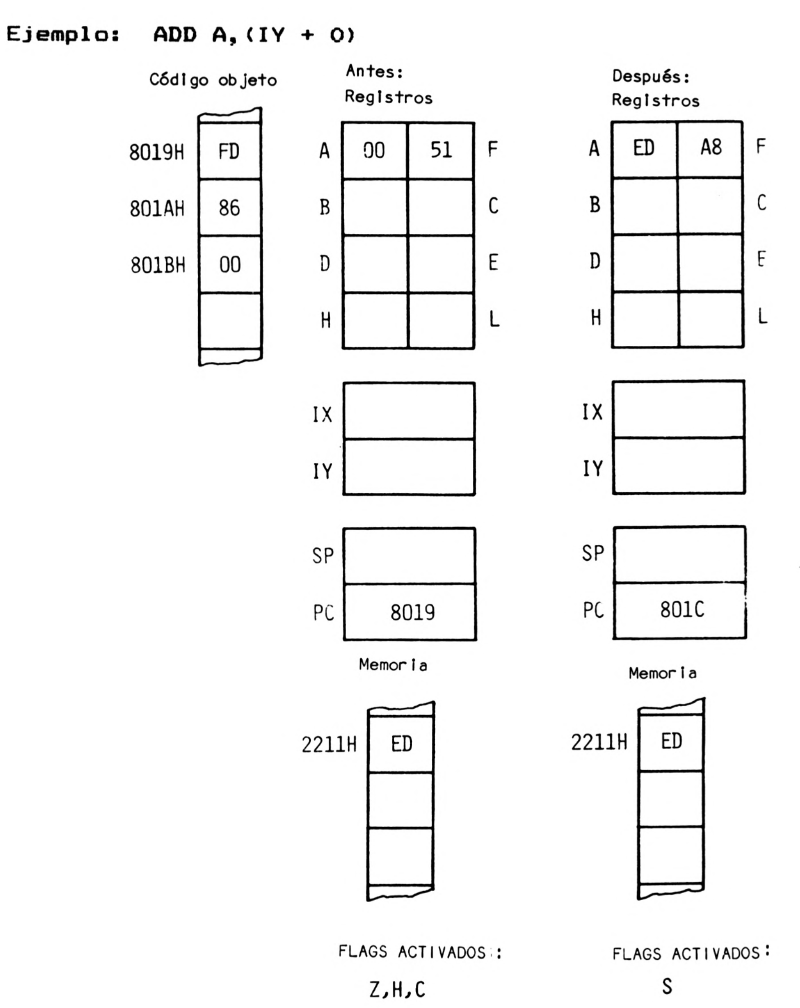
Direccionamiento: Indexado

Formato:

Byte 1)	1	1	1	1	1	1	0	1	FDH
Byte 2)	1	0	0	0	0	1	1	0	86H

Byte 3) -----------d----------Offset

Ejemplo: ADD A,(IY + 0)



ADD A,n — Suma el acumulador con los datos inmediatos n.

Función: A <—— A + n

Flags:

S	Z	-	H	-	P/V	N	C
X	X		X		X	0	X

Descripción: Al contenido del acumulador A se sumará el dato n de 8 bits. El resultado se almacenará en el acumulador.

Ejecución:

Tiempo de ejecución	: 1,75 µs
Frecuencia de reloj	: 4,00 MHz
Estados de reloj	: 7
Ciclos de máquina	: 2

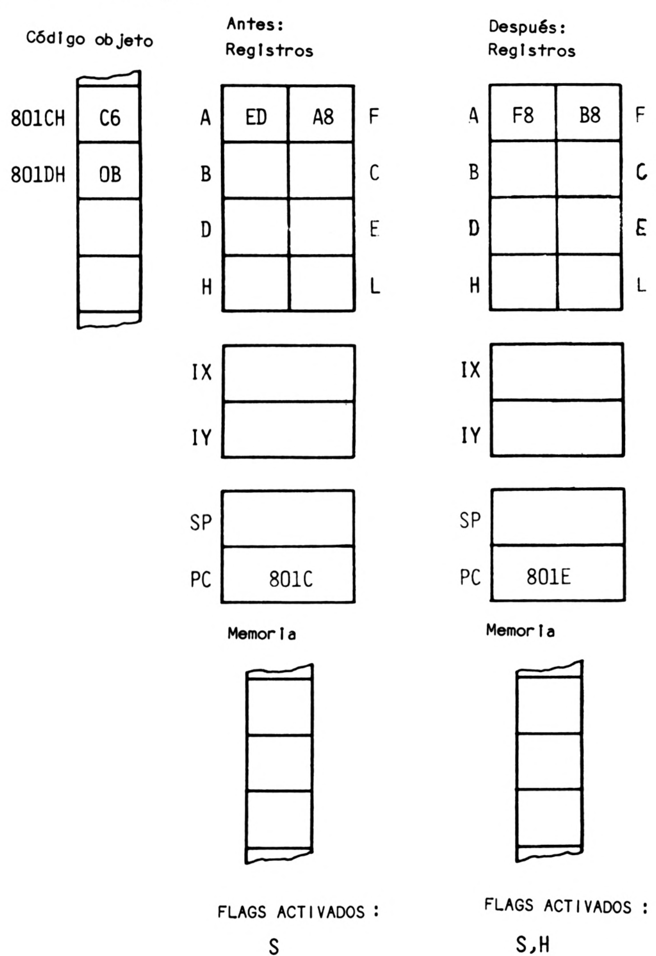
Direccionamiento: Inmediato

Formato:

Byte 1) 1 1 0 0 0 1 1 0 C6H

Byte 2) -----------n----------datos inmed.

Ejemplo: ADD A,11



<u>ADD A,r</u> Suma el acumulador con el registro r

Función: A <——— A + r

Flags:

S	Z	-	H	-	P/V	N	C
X	X		X		X	O	X

Descripción: Al contenido del acumulador A se sumará el contenido del registro r. El resultado se almacenará en el HL.

Ejecución:

Tiempo de ejecución	:	1,00 µs
Frecuencia de reloj	:	4,00 MHz
Estados de reloj	:	4
Ciclos de máquina	:	1

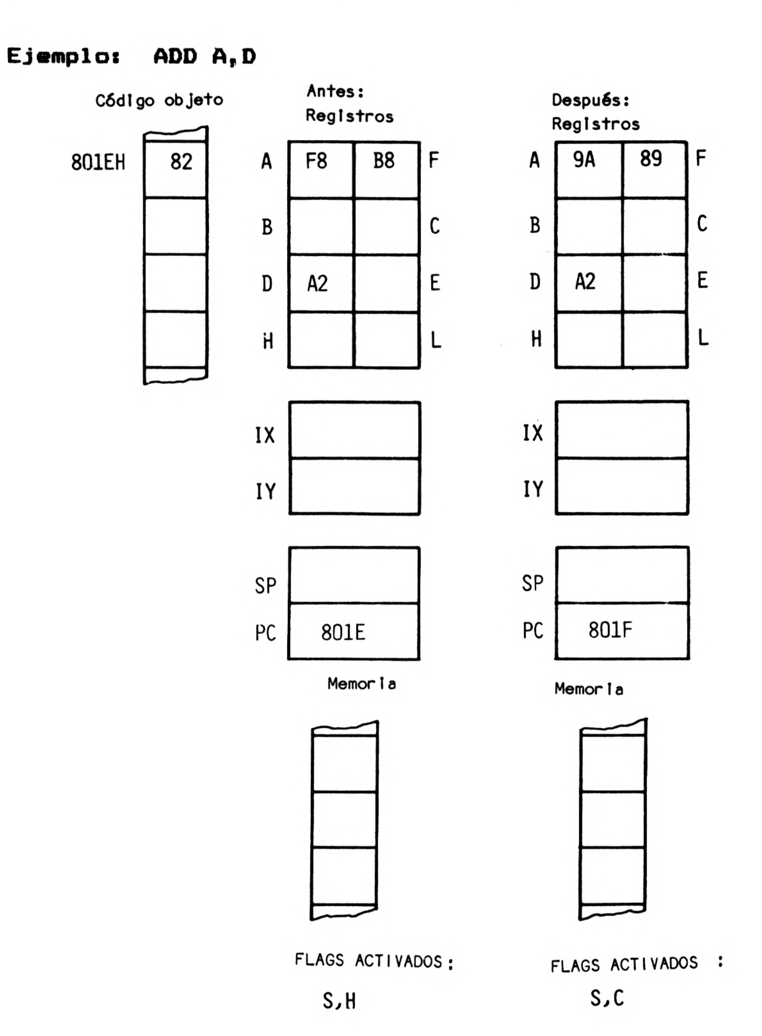
Direccionamiento: Implícito

Formato: 1 0 0 0 0 --r--

r puede ser:

A - 111	E - 011
B - 000	H - 100
C - 001	L - 101
D - 010	

Ejemplo: ADD A,D



ADD HL,ss — Suma HL y el par de registros ss.

Función: HL <——— HL + ss

Flags:

S	Z	-	H	-	P/V	N	C
			X			0	X

Descripción: Al par de registros HL se sumará el contenido del par de registros ss. El resultado se almacenará en el HL.

Ejecución:

Tiempo de ejecución	: 2,75 µs
Frecuencia de reloj	: 4,00 MHz
Estado de reloj	: 11
Ciclos de máquina	: 3

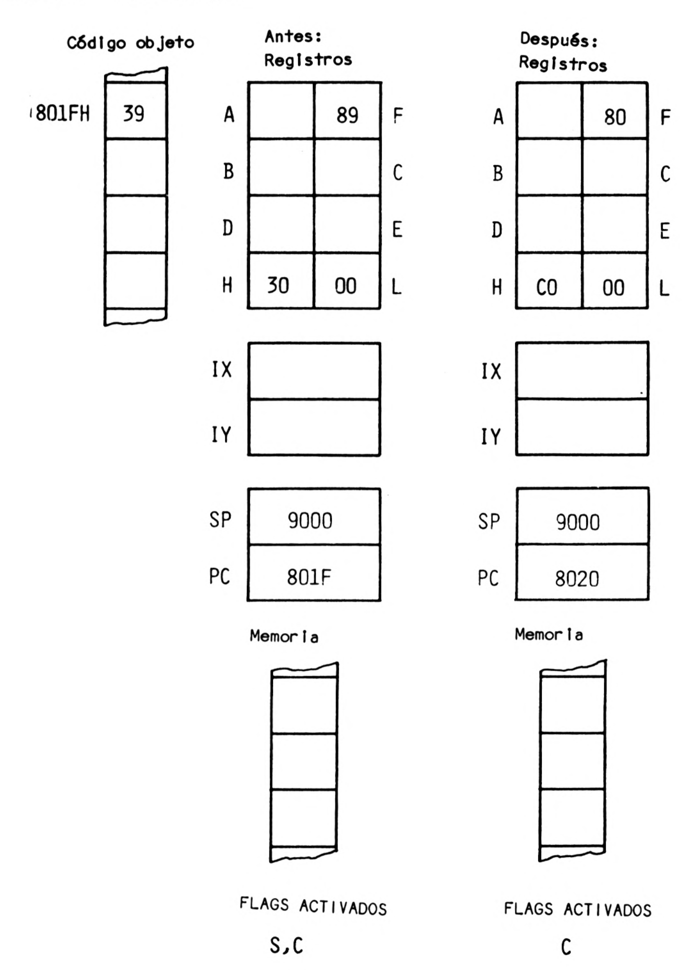
Direccionamiento: Implícito

Formato: 0 0 s s 1 0 0 1

ss puede ser:

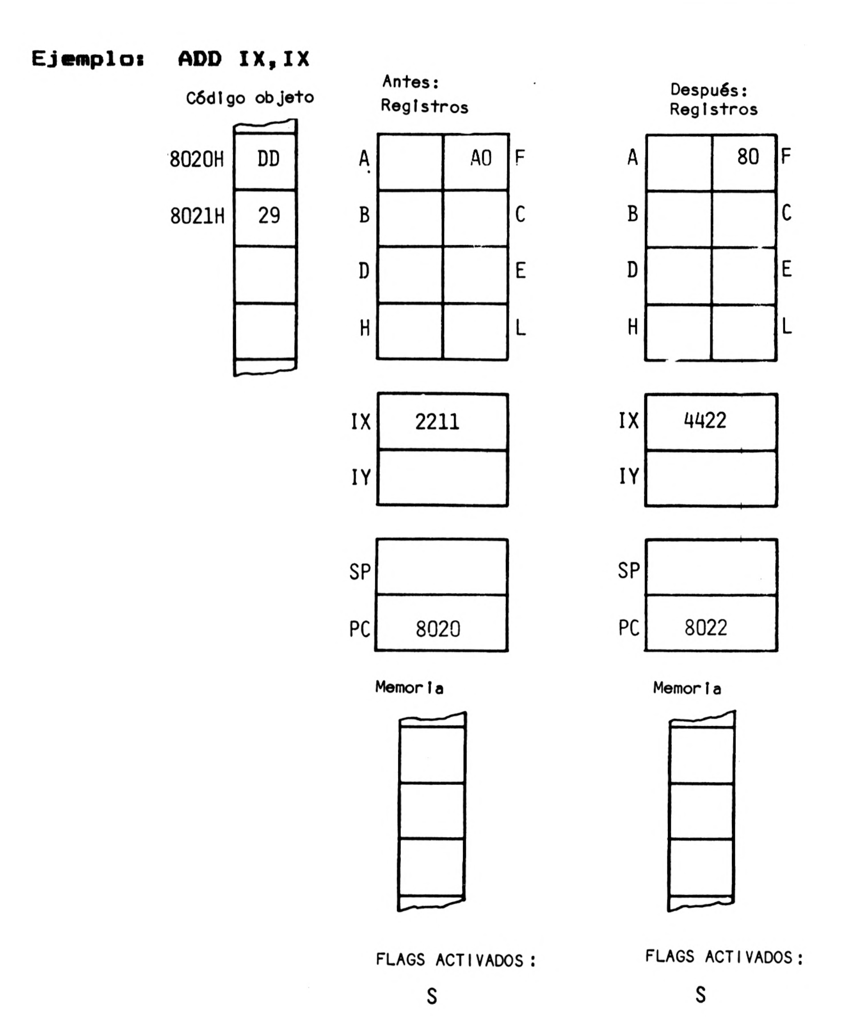
BC - 00	HL - 10
DE - 01	SP - 11

Ejemplo: ADD HL,SP



ADD IX,rr	Suma IX con el par de registros rr.
Función:	IX <——— IX + rr
Flags:	S Z - H - P/V N C X 0 X
	H se activa mediante acarreo del bit 11. C se activa mediante acarreo del bit 15.
Descripción:	Al contenido del par de registros IX se sumará el contenido del par de registros rr. El resultado se almacenará en el par de registros IX.
Ejecución:	Tiempo de ejecución : 3,75 µs Frecuencia de reloj : 4,00 MHz Estados de reloj : 15 Ciclos de máquina : 4
Direccionamiento:	Implícito
Formato:	Byte 1) 1 1 0 1 1 1 0 1 DDH Byte 2) 0 0 r r 1 0 0 1
	rr puede ser: BC - 00 IX - 10 DE - 01 SP - 11

Ejemplo: ADD IX,IX



ADD IY,rr — Suma IY y el par de registros rr.

Función: IY <——— IY + rr

Flags:

S	Z	-	H	-	P/V	N	C
			X			0	X

H se activa mediante acarreo del bit 11.
C se activa mediante acarreo del bit 15.

Descripción: Al contenido del par de registros IY se sumará el contenido del par de registros rr. El resultado se almacenará en el par de registros IY.

Ejecución:

Tiempo de ejecución : 3,75 µs
Frecuencia de reloj : 4,00 MHz
Estados de reloj : 15
Ciclos de máquina : 4

Direccionamiento: Implícito

Formato:

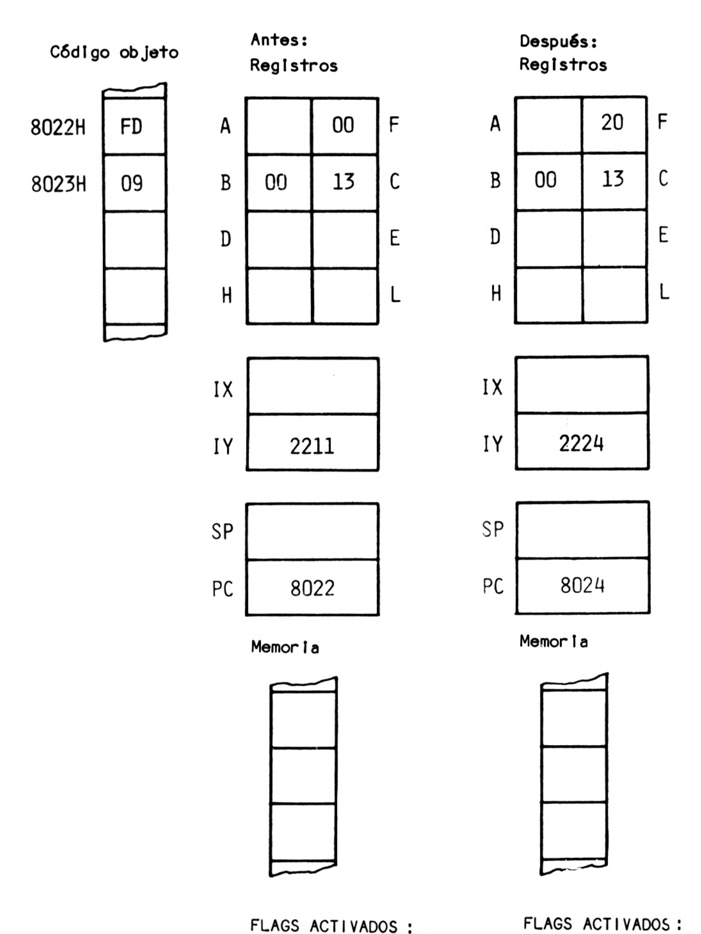
Byte 1)	1	1	1	1	1	1	0	1	FDH
Byte 2)	0	0	r	r	1	0	0	1	

rr puede ser:

BC - 00 IY - 10

DE - 01 SP - 11

Ejemplo: ADD IY,BC



AND s

El acumulador y el operando s se relacionan mediante la función lógica "AND".

Función: A <——— A AND s

Flags:

S	Z	-	H	-	P/V	N	C
X	X		1		X	0	0

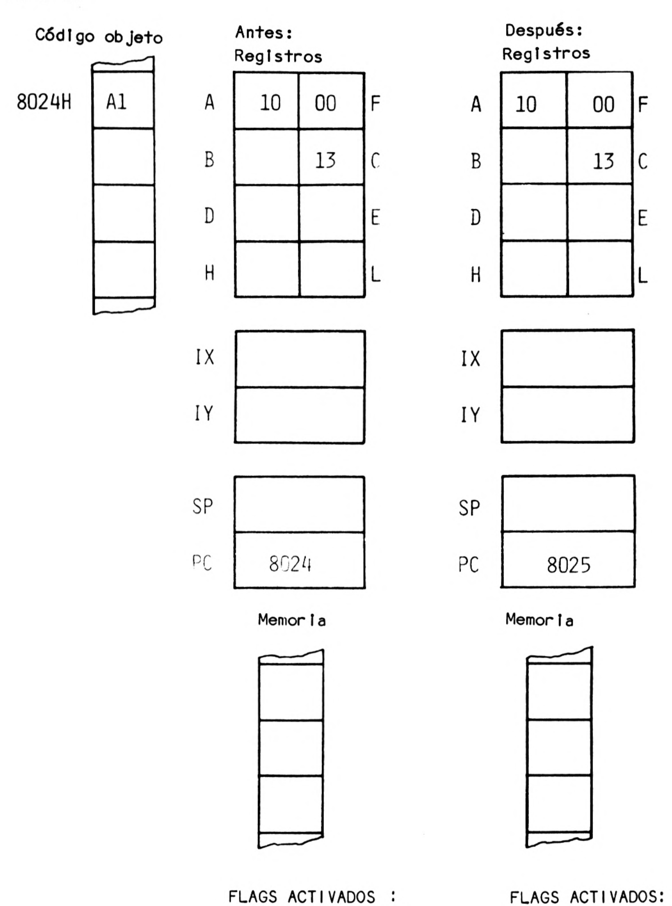
Descripción: El acumulador A y el operando s se relacionan mediante "AND" lógico. El resultado será almacenado en el acumulador.

Ejecución:

s	µs (4,00 MHz)	Estados T	Ciclos M
r	1,00	4	1
n	1,75	7	2
(HL)	1,75	7	2
(IX+d)	4,75	19	5
(IY+d)	4,75	19	5

Direccionamiento: r: implícito/ n: inmediato/ (HL): indirecto/ (IX+d),(IX+d): indexado

Ejemplo: AND C



BIT b,(HL) — Prueba el bit b de la célula de memoria (HL) indirectamente direccionada.

Función: $Z \longleftarrow (HL)_b$

Flags:

S	Z	-	H	P/V	N	C
?	X		1	?	0	

Descripción: El bit b del operando (HL) direccionado indirectamente a través del par de registros HL será probado en cuanto a su estado. El resultado influirá el zeroflag Z. Al zeroflag se le asignará el estado contrario del bit probado.

Ejecución:

Tiempo de ejecución : 3 µs
Frecuencia de reloj : 4 MHz
Estado de reloj : 12
Ciclos de máquina : 3

Direccionamiento: Indirecto

Formato:

Byte 1) 1 1 0 0 1 0 1 1 CBH

Byte 2) 0 1 ----b-----1 1 0

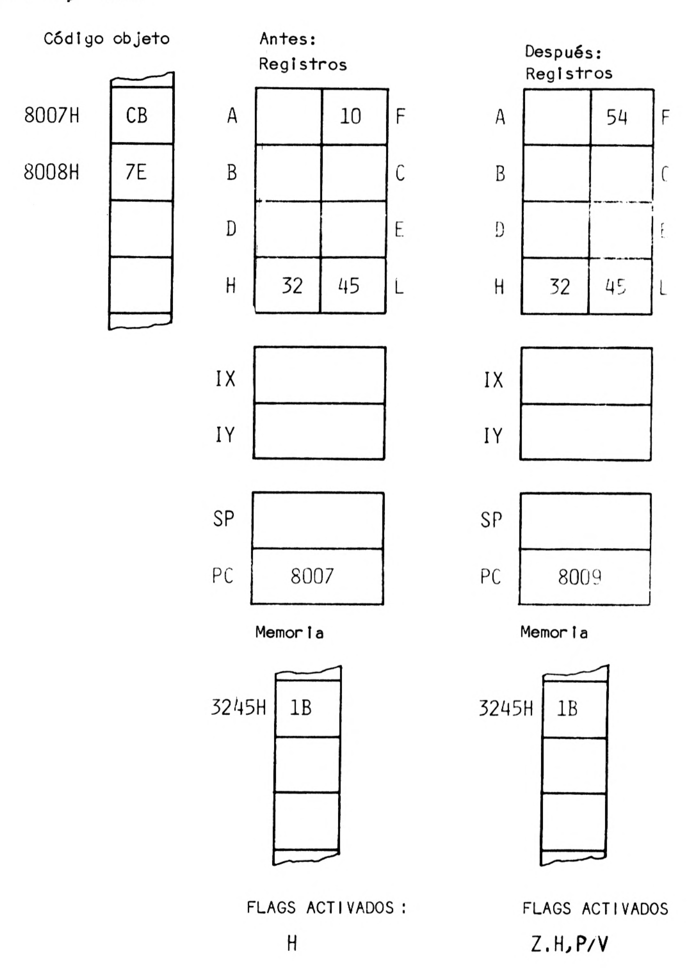
b puede ser:

0 - 000	4 - 100
1 - 001	5 - 101
2 - 010	6 - 110
3 - 011	7 - 111

b puede ser:

0 - 000	5 - 101
1 - 001	6 - 110
2 - 010	7 - 111
3 - 011	
4 - 100	

Ejemplo: BIT 7, (HL)



BIT b,(IX + d)	Prueba el bit b de la célula de memoria direccionada indexadamente por (IX + d).
Función:	Z <——— $(IX + d)_b$
Flags:	S Z - H - P/V N C ? X 1 ? 0
Descripción:	Al bit b del operando (IX + d) direccionado indexadamente a través del par de registros IX, y del desplazamiento d, se le asignará el estado invertido al del bit probado.
Ejecución:	Tiempo de ejecución : 5,00 µs Frecuencia de reloj : 4 MHz Estados de reloj : 20 Ciclos de máquina : 5
Direccionamiento:	Indexado
Formato:	Byte 1) 1 1 0 1 1 1 0 1 DDH Byte 2) 1 1 0 0 1 0 1 1 CBH Byte 3) ----------d-----------Offset Byte 4) 0 1----b------1 1 0

b puede ser:

0 - 000

1 - 001

2 - 010

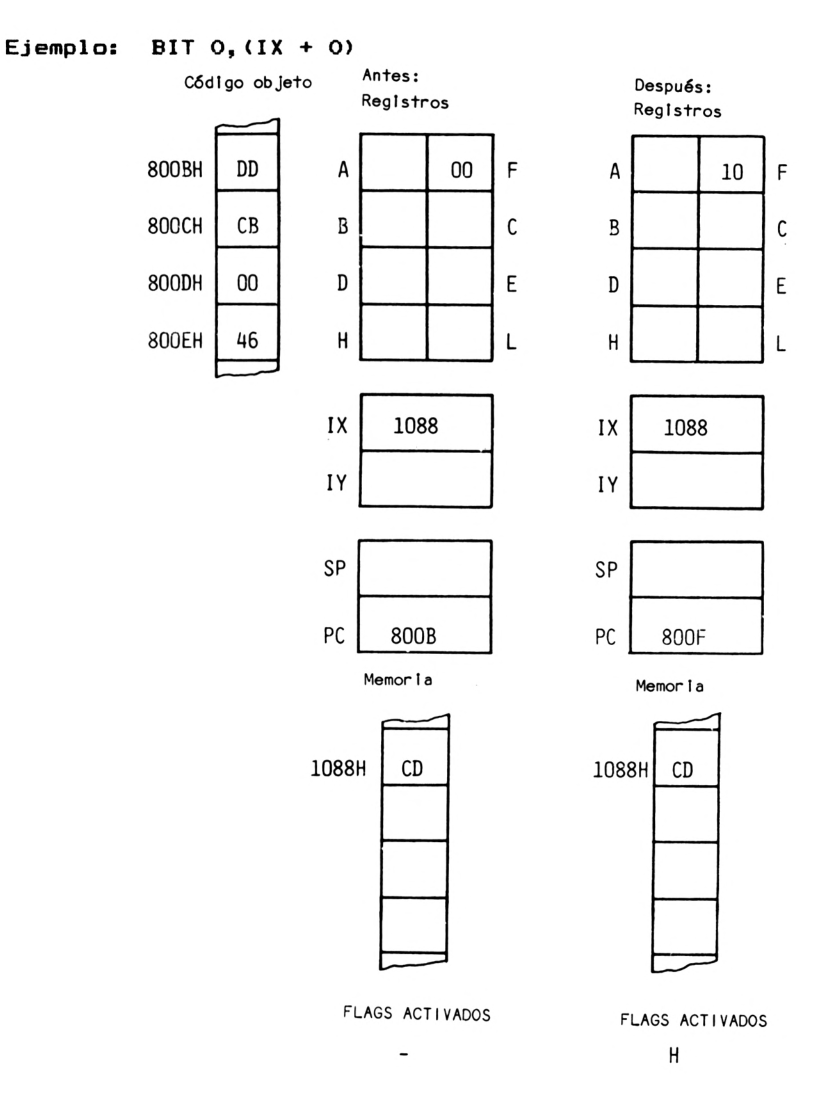
3 - 011

4 - 100

5 - 101

6 - 110

Ejemplo: **BIT 0,(IX + 0)**



BIT b,(IY + d) Prueba el bit b de la célula de memoria (IY + d) direccionada indexadamente.

Función: Z <——— $(IY + d)_b$

Flags:

S	Z	-	H	-	P/V	N	C
?	X		1		?	0	

Descripción: Al bit b del operando (IY + d) direccionado indexadamente a través del par de registros IY, y del desplazamiento d, se le asignará el estado invertido al del bit probado.

Ejecución:

Tiempo de ejecución	: 5,00 µs
Frecuencia de reloj	: 4,00 MHz
Estados de reloj	: 20
Ciclos de máquina	: 5

Direccionamiento: Indexado

Formato:

Byte 1) 1 1 1 1 1 1 0 1 FDH

Byte 2) 1 1 0 0 1 0 1 1 CBH

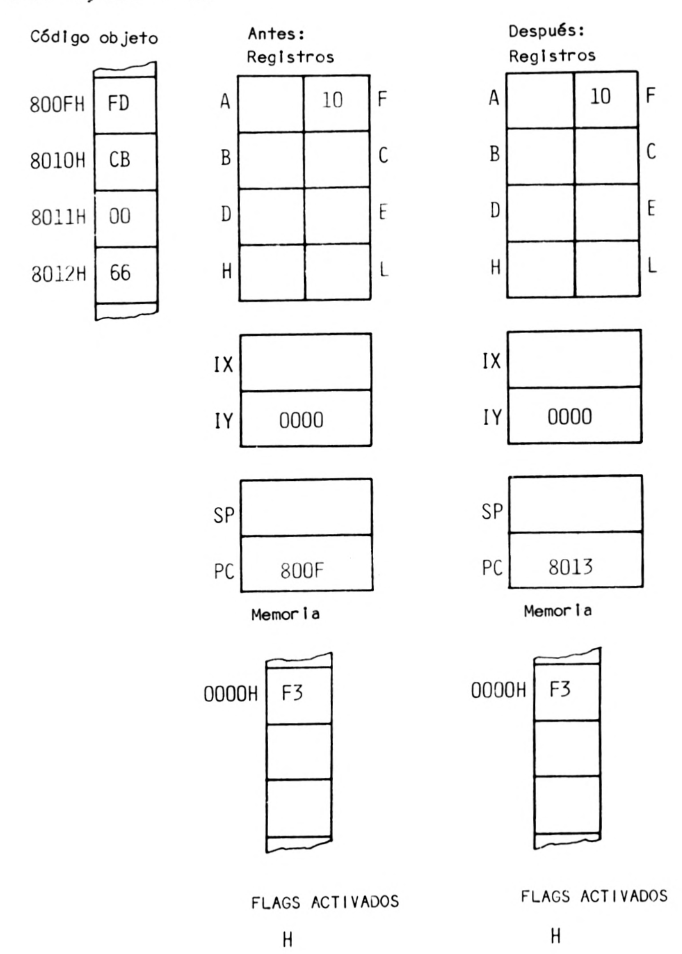
Byte 3) -----------d----------Offset

Byte 4) 0 1-----b-----1 1 0

b puede ser:

0 - 000	4 - 100
1 - 001	5 - 101
2 - 010	6 - 110
3 - 011	

Ejemplo: BIT 4,(IY + 0)



BIT b,r Prueba el bit b del registro r.

Función: Z <——— r_b

Flags:

S	Z	-	H	-	P/V	N	C
?	X		1		?	0	

Descripción: El bit b del registro r será probado con respecto a su estado. El resultado influenciará el zeroflag Z. Al zeroflag se le asignará el estado invertido al del bit probado.

Ejecución:

Tiempo de ejecución	: 2,00 µs
Frecuencia de reloj	: 4,00 MHz
Estados de reloj	: 8
Ciclos de máquina	: 2

Direccionamiento: Implícito

Formato: Byte 1) 1 1 0 0 1 0 1 1 CBH

Byte 2) 0 1-----b--------r---

b y r pueden ser:

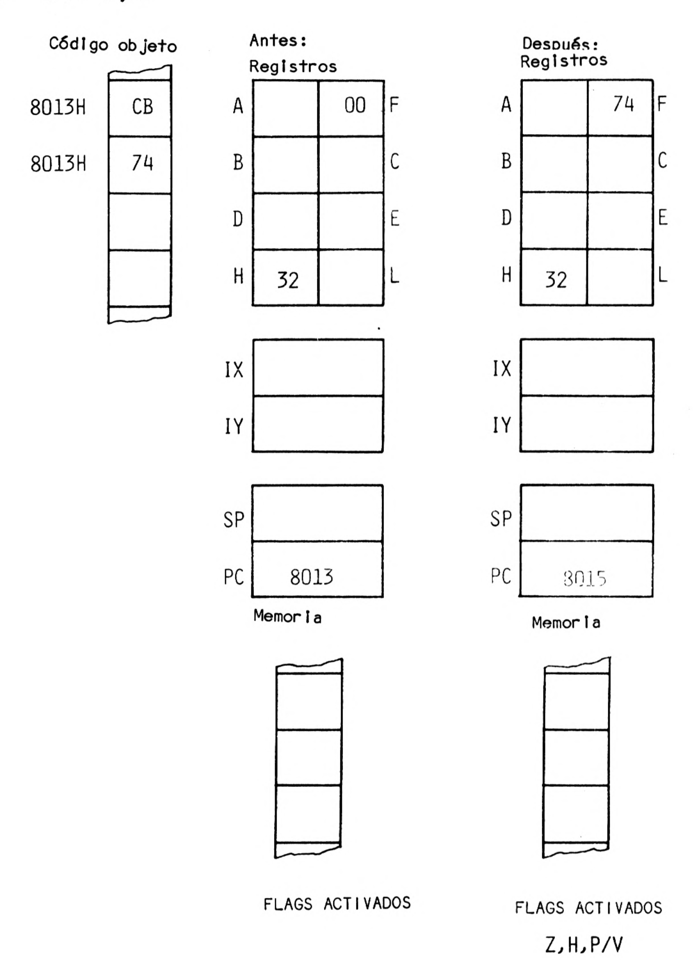
b:

0 - 000	4 - 100
1 - 001	5 - 101
2 - 010	6 - 110
3 - 011	7 - 111

r:

A - 111	E - 011
B - 000	H - 100
C - 001	L - 101

Ejemplo: BIT 6,H



Direccionamiento: Inmediato

Formato:

Byte 1) 1 1 ---cc---1 0 0

Byte 2) ----------q---------Posición inferior direc.

Byte 3) ----------p---------Posición superior direc.

cc puede ser:

NZ	-	000	PO	-	100
Z	-	001	PE	-	101
NC	-	010	P	-	100
C	-	011	M	-	111

CALL cc,pq — Llamada condicional del subprograma

Función: Cuando el cc se haya realizado (SP − 1) -- PC_{arriba}; (SP − 2) -- PC_{abajo}; SP -- SP − 2; PC -- pq cuando el cc no se haya realizado: PC -- PC + 3

Flags: S Z − H − P/V N C
(ninguna influencia)

Descripción: El contenido del apuntador de pila SP direcciona en primer lugar la última entrada en la pila. El apuntador de pila SP disminuirá, cargándose así la posición de memoria direccionada en la pila con el byte de mayor valor del contador del programa PC. Después de decrementar nuevamente el apuntador de pila SP se cargará el byte de inferior valor del contador del programa PC en la posición de memoria direccionada en la pila a través del apuntador de pila SP. Después se cargará la q en la parte inferior y la p en la parte superior del PC. El programa continuará a partir de la dirección pq. Mediante la instrucción RET podrá volverse a recuperar de la pila el antiguo contenido del contador del programa PC al final del subprograma. El programa principal continuará nuevamente con su trabajo a partir de la dirección que se encuentra después de la instrucción Call. Es decir cuando se haya cumplido la condición cc. Si cc no se cumple, no se bifurcará al subprograma sino que se continuará con la siguiente instrucción.

Ejecución:

	μs (4,00 MHz)	Estados T	Ciclos M
Cumple condición	4,25	17	5
No cumple condición	2,50	10	3

Formato:

Byte 1) 1 1 - cc - 1 0 0

Byte 2) -----------q-----------Posición inferior direc.

Byte 3) -----------p-----------Posición superior direc.

cc puede ser:

NZ - 000 PO - 100

Z - 001 PE - 101

NC - 010 P - 100

C - 011 M - 111

CALL pq — Llamada de un subprograma en la dirección pq.

Función: (SP - 1) -- PC_{arriba}; (SP - 2) -- PC_{abajo}; SP--SP - 2; PC -- PQ

Flags: S Z - H - P/V N C
(ninguna influencia)

Descripción: El contenido del apuntador de pila SP direccionará en primer lugar la última entrada en la pila. El apuntador de pila SP se decrementará, cargándose la posición de memoria direccionada en la pila con el byte de mayor valor del contador del programa PC. Después de decrementar nuevamente el apuntador de pila SP, se cargará en la pila el byte de inferior valor del contador del programa PC en la posición de la memoria direccionada mediante el apuntador de pila SP. Después se cargará la q en la parte inferior y la p en la parte superior del PC. El programa continuará su ejecución a partir de la dirección pq. Mediante la instrucción RET podrá recuperarse de la pila el antiguo contenido del contador del programa PC al final del subprograma. El programa principal continuará nuevamente con su trabajo a partir de la dirección que se encuentra después de la instrucción call.

Ejecución:

Tiempo de ejecución	:	4,25 µs
Frecuencia de reloj	:	4,00 MHz
Estados de reloj	:	17
Ciclos de máquina	:	5

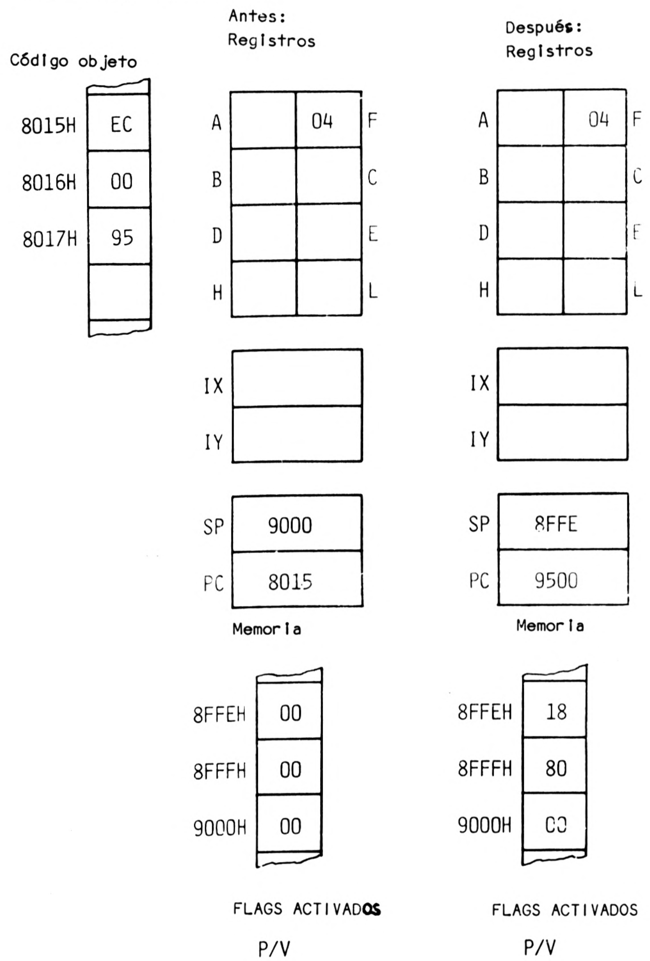
```
Formato:             Byte 1) 1  1  0  0  1  1  0  1 CDH

                     Byte 2) -----------q----------Mitad
                                                   infer.
                                                   direc.
                     Byte 3) -----------p----------Mitad
                                                   super.
                                                   direc.

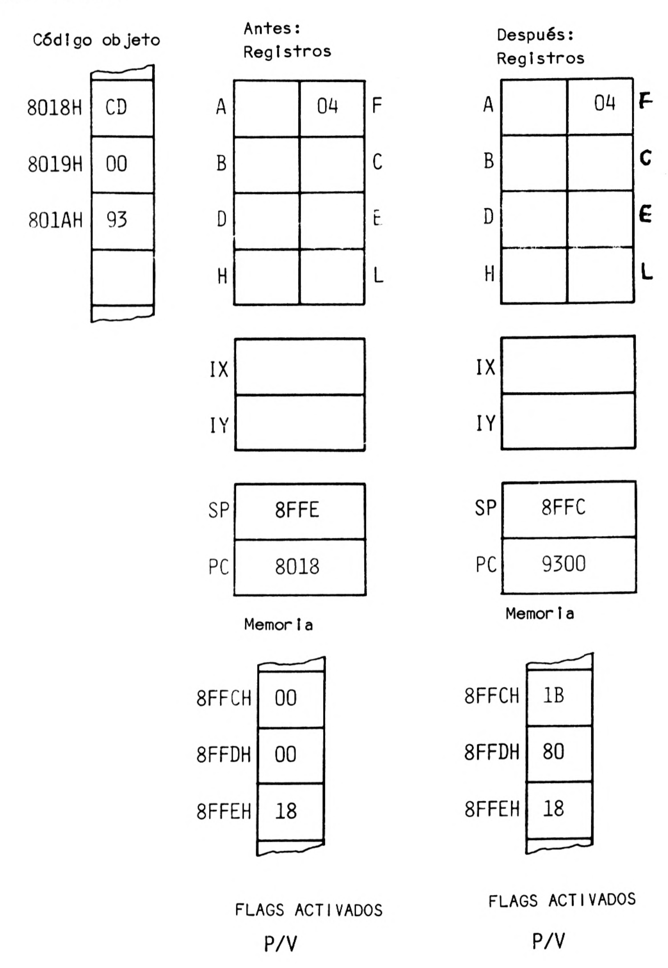

                     Byte 1) 1  1  0  0  1  1  0  1 CDH

                     Byte 2) -----------q----------Mitad
                                                   infer.
                                                   direc.
                     Byte 3) -----------p----------Mitad
                                                   super.
                                                   direc.
```

Ejemplo: CALL PE, 9500H



Ejemplo: CALL 9300H



CCF — **Complemento del flag de acarreo.**

Función: C <——— C

Flags:

S	Z	-	H	-	P/V	N	C
			X			0	X

Contiene el estado precedente del flag de acarreo.

Descripción: Al carryflag se le asignará su valor complementario. Si previamente tenía el valor 1, trás la ejecución de la instrucción habrá un 0 en el carryflag y viceversa.

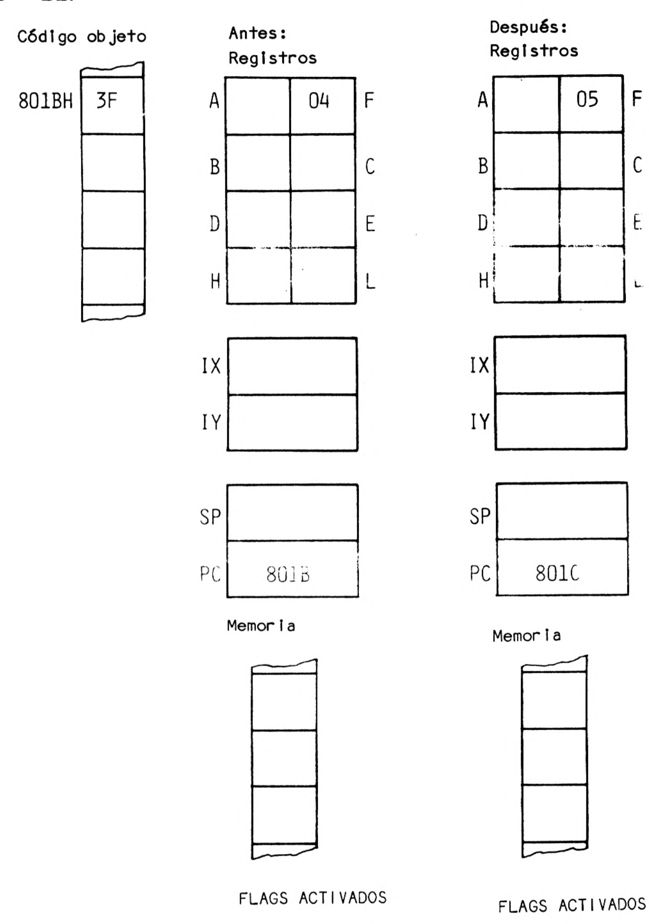
Ejecución:

Tiempo de ejecución	: 1,00 µs
Frecuencia de reloj	: 4,00 MHz
Estados de reloj	: 4
Ciclos de máquina	: 1

Direccionamiento: Implícito

Formato: 0 0 1 1 1 1 1 1 3FH

Ejemplo: CCF



FLAGS ACTIVADOS

P/V

FLAGS ACTIVADOS

P/V,C

CP s — Comparar el operando s con el acumulador.

Función: A − s

Flags:

S	Z	-	H	-	P/V	N	C
X	X		X		X	1	X

Descripción: Del acumulador A se restará el operando s. El acumulador mantendrá su valor original, ya que el resultado únicamente se utiliza para influenciar a los flags.

Ejecución:

s	µs (4,00 MHz)	Estados T	Ciclos M
r	1,00	4	1
n	1,75	7	2
(HL)	1,75	7	2
(IX+d)	4,75	19	5
(IY+d)	4,75	19	5

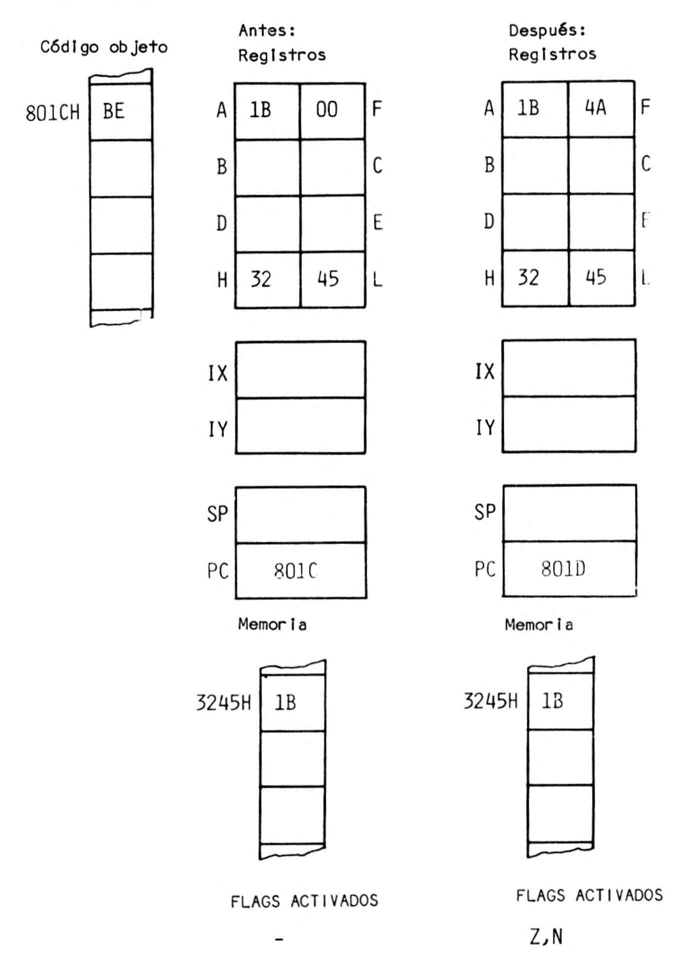
Direccionamiento: r: implícito/ n: inmediato/ (HL): indirecto/ (IX+d),(IY+d): indexado

Formato:	r:		1 0 1 1 1---r--
	n:		1 1 1 1 1 1 1 0 FEH
		Byte 2)	------n--------Fecha
	(HL):	Byte 1)	1 0 1 1 1 1 1 0 BEH
	(IX+d):	Byte 1)	1 1 0 1 1 1 0 1 DDH
		Byte 2)	1 0 1 1 1 1 1 0 BEH
		Byte 3)	------d--------Offset
	(IY+d):	Byte 1)	1 1 1 1 1 1 0 1 FDH
		Byte 2)	1 0 1 1 1 1 1 0 BEH
		Byte 3)	------d--------Offset

r puede ser:

A - 111	E - 011
B - 000	H - 100
C - 001	L - 101
D - 010	

Ejemplo: CP (HL)



CPD Compara y decrementa.

Función: A - (HL); HL <— HL - 1 BC <— BC - 1

Flags:

S	Z	-	H	-	P/V	N	C
X	X		X		X	1	

Descripción: Del acumulador A se restará el operando (HL) direccionado indirectamente a través del par de registros HL. El acumulador mantendrá, sin embargo, su valor original, ya que el resultado se utiliza únicamente para influenciar a los flags. El BC sirve como contador de bucles, el cual, tal como ocurre en el par de registros HL, será finalmente decrementado. Si el BC = 0, el zeroflag será asímismo 0; de lo contrario será 1. El flag de paridad se activa en 1, cuando exista igualdad entre el contenido del acumulador y el del operando (HL).

Ejecución:

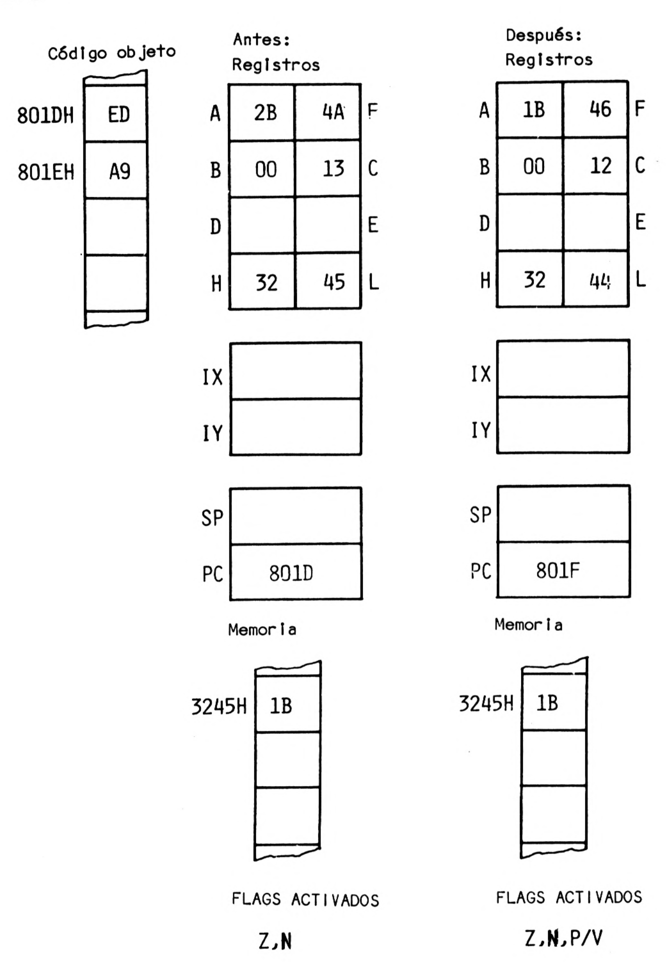
Tiempo de ejecución : 4,00 µs
Frecuencia de reloj : 4,00 MHz
Estados de reloj : 16
Ciclos de máquina : 4

Direccionamiento: Indirecto

Formato:

Byte 1)	1	1	1	0	1	1	0	1	EDH
Byte 2)	1	0	1	0	1	0	0	1	A9H

Ejemplo: CPD


Código objeto
801DH
ED
801EH
A9
Antes:
Registros
A
2B
4A
F
B
00
13
C
D
E
H
32
45
L
IX
IY
SP
PC
801D
Memoria
3245H
1B
FLAGS ACTIVADOS
Z,N
Después:
Registros
A
1B
46
F
B
00
12
C
D
E
H
32
44
L
IX
IY
SP
PC
801F
Memoria
3245H
1B
FLAGS ACTIVADOS
Z,N,P/V

CPDR — Comparación de bloque y decremento.

Función: A - (HL); HL <— HL - 1; BC <— BC-1;
Repite hasta BC = 0 ó A = (HL)

Flags:

S	Z	-	H	-	P/V	N	C
X	X		X		X	1	

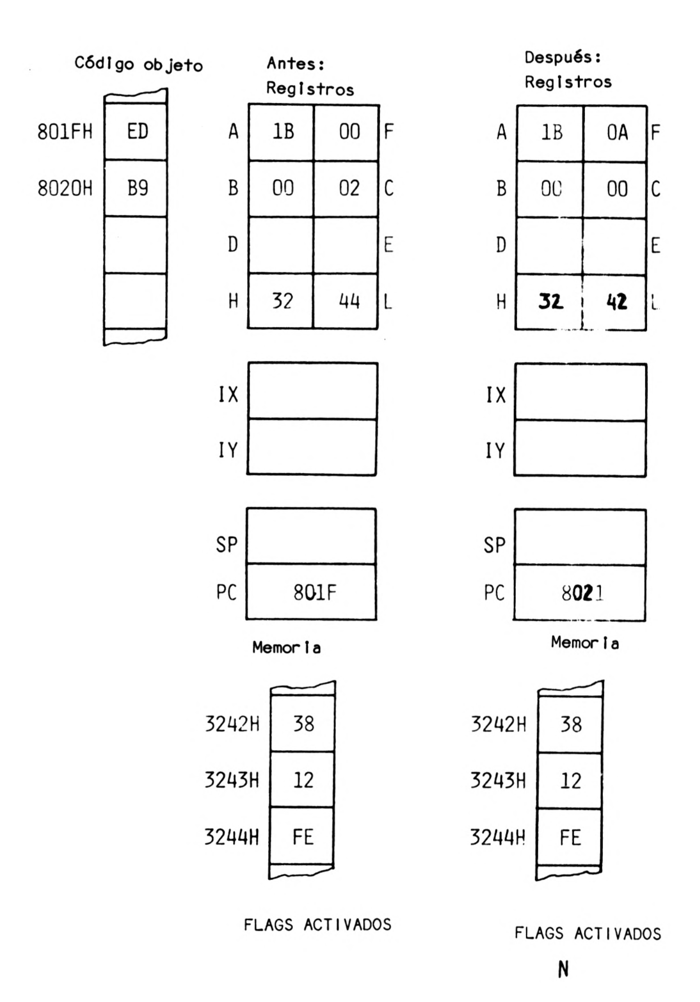
Descripción: Del acumulador A se restará el operando (HL) direccionado indirectamente a través del par de registros HL. El acumulador mantendrá sin embargo su valor original, ya que el resultado se utiliza únicamente para influenciar los flags. El BC sirve como contador de bucles, el cual, tal como ocurre en el par de registros HL, será finalmente decrementado. Si el BC = 0, el zeroflag será asimismo 0; de lo contrario será 1. El flag de paridad se activa en 1, cuando exista igualdad entre el contenido del acumulador y el del operando (HL). Mientras el zeroflag sea diferente de 1 y el flag de paridad sea 1, el contador del programa PC se decrementará en 2 y la instrucción volverá a ejecutarse.

Ejecución: BC = 0 ó A = HL : 4 ciclos M; 16 estados T: 4 µs 4 MHz
BC = 0 y A = HL : 5 ciclos M; 21 estados T: 5,25 µs 4 MHz

Formato:

Byte 1)	1	1	1	0	1	1	0	1	EDH
Byte 2)	1	0	1	1	1	0	0	1	B9H

Ejemplo: CPDR



CPI Compara e incrementa.

Función: A <— (HL); HL <— HL+1; BC <— BC -1

Flags:

S	Z	-	H	-	P/V	N	C
X	X		X		X	1	

P/V: Desactivado si después de la ejecución BC = 0, de lo contrario permanece activado.
Z: Activado cuando A = HL.

Descripción: Del acumulador A se restará el operando (HL) direccionado indirectamente a través del par de registros HL. El acumulador mantendrá sin embargo su valor original, ya que el resultado se utiliza únicamente para influenciar los flags. El BC sirve como contador de bucles, el cual, tal como ocurre con el par de registro HL, será finalmente decrementado. Si el BC = 0, el zeroflag será asímismo 0; de lo contrario será 1. El flag de paridad se activa en 1, cuando exista igualdad entre el contenido del acumulador y el del operando (HL).

Ejecución:

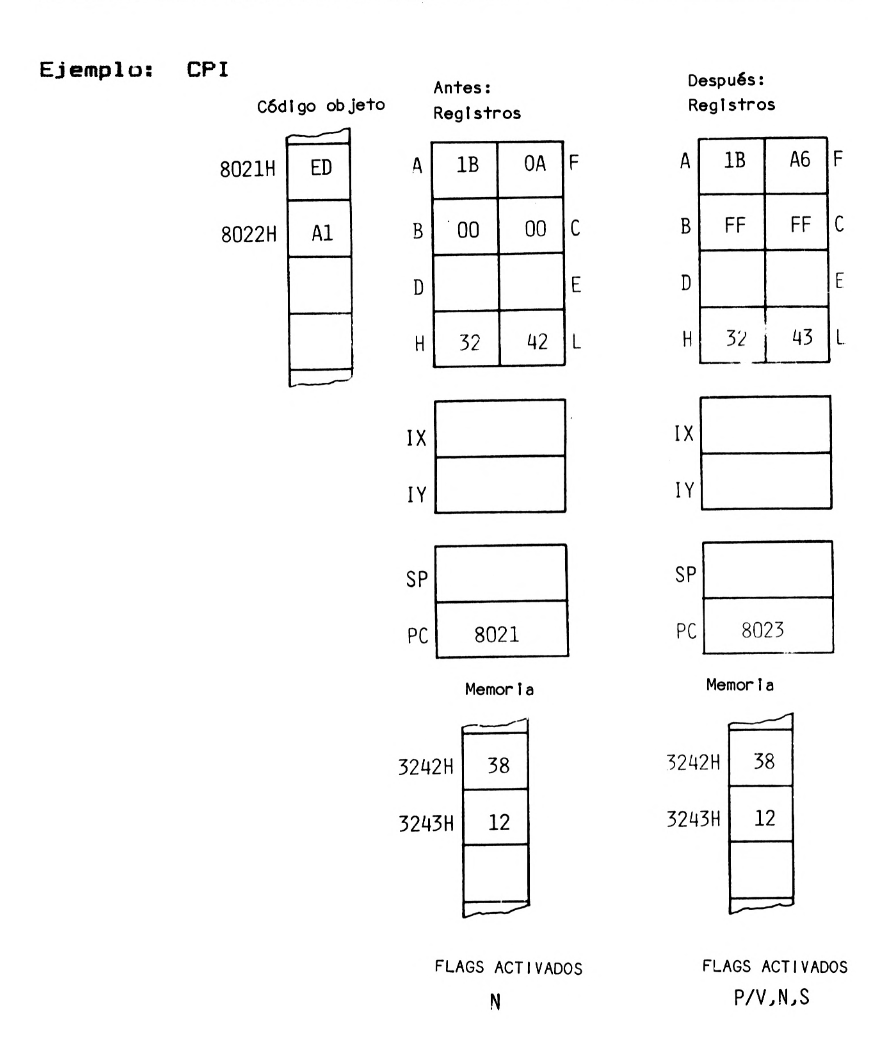
Tiempo de ejecución	:	4,00 µs
Frecuencia de reloj	:	4,00 MHz
Estados de reloj	:	16
Ciclos de máquina	:	4

Direccionamiento: Indirecto

Formato:

Byte 1)	1	1	1	0	1	1	0	1	EDH
Byte 2)	1	0	1	0	0	0	0	1	A1H

Ejemplo: CPI


Código objeto
8021H
ED
8022H
A1
Antes:
Registros
A
1B
0A
F
B
00
00
C
D
E
H
32
42
L
IX
IY
SP
PC
8021
Memoria
3242H
38
3243H
12
FLAGS ACTIVADOS
N
Después:
Registros
A
1B
A6
F
B
FF
FF
C
D
E
H
32
43
L
IX
IY
SP
PC
8023
Memoria
3242H
38
3243H
12
FLAGS ACTIVADOS
P/V,N,S

CPIR — Comparación de bloques e incremento.

Función: A - (HL); HL <—— HL+1; BC <—— BC-1; Repite hasta BC = 0 ó A = (HL)

Flags: S Z - H - P/V N C
Desactivado si después de la ejecución BC = 0, de lo contrario queda activado.
Cuando A = HL se activa.

Descripción: Del acumulador A se restará el operando (HL) direccionado indirectamente a través del par de registros HL. El acumulador mantendrá sin embargo su valor original, ya que el resultado se utiliza únicamente para influenciar a los flags. El BC sirve como contador de bucles, el cual, tal como ocurre con el par de registros HL, será finalmente decrementado. Si el BC = 0, el zeroflag será asimismo 0; de lo contrario será 1. El flag de paridad se activa en 1, cuando exista igualdad entre el contenido del acumulador y del operando (HL). Mientras el zeroflag no sea 1 y el flag de paridad sea 1 el contador del programa PC se incrementará en 2 y la instrucción será nuevamente ejecutada.

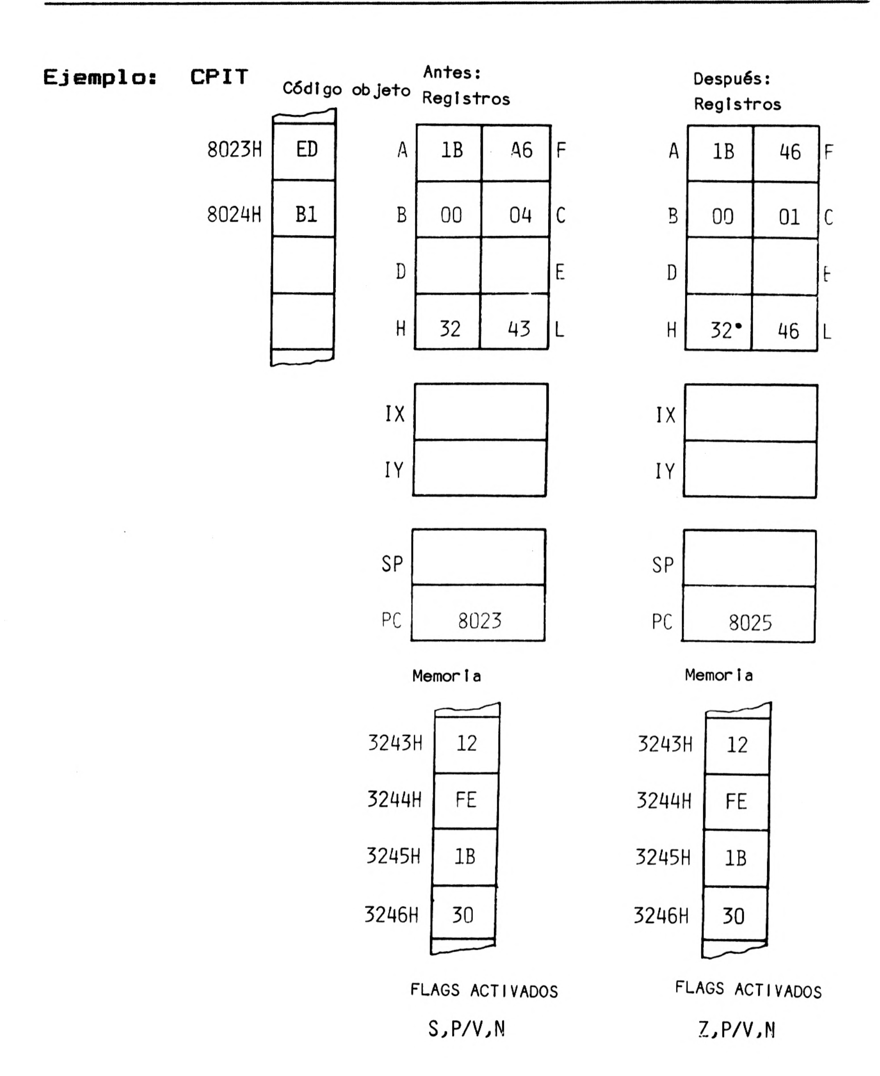
Ejecución: BC = 0 ó A = HL; ciclos M 4; estado T 16: 4 µs 4 MHz
BC = 0 y A = HL; ciclos M 5; estado T 21: 5,25 µs 4 MHz

Direccionamiento: Indirecto

Formato:

Byte 1)	1	1	1	0	1	1	0	1	EDH
Byte 2)	1	0	1	1	0	0	0	1	B1H

Ejemplo: CPIT



CPL　　Complementa el acumulador

Función:　　A <——— A

Flags:

S	Z	-	H	-	P/V	N	C
			1			1	

Descripción:　　El acumulador será complementado, es decir, que todos los ceros del acumulador se convertirán en unos y viceversa.

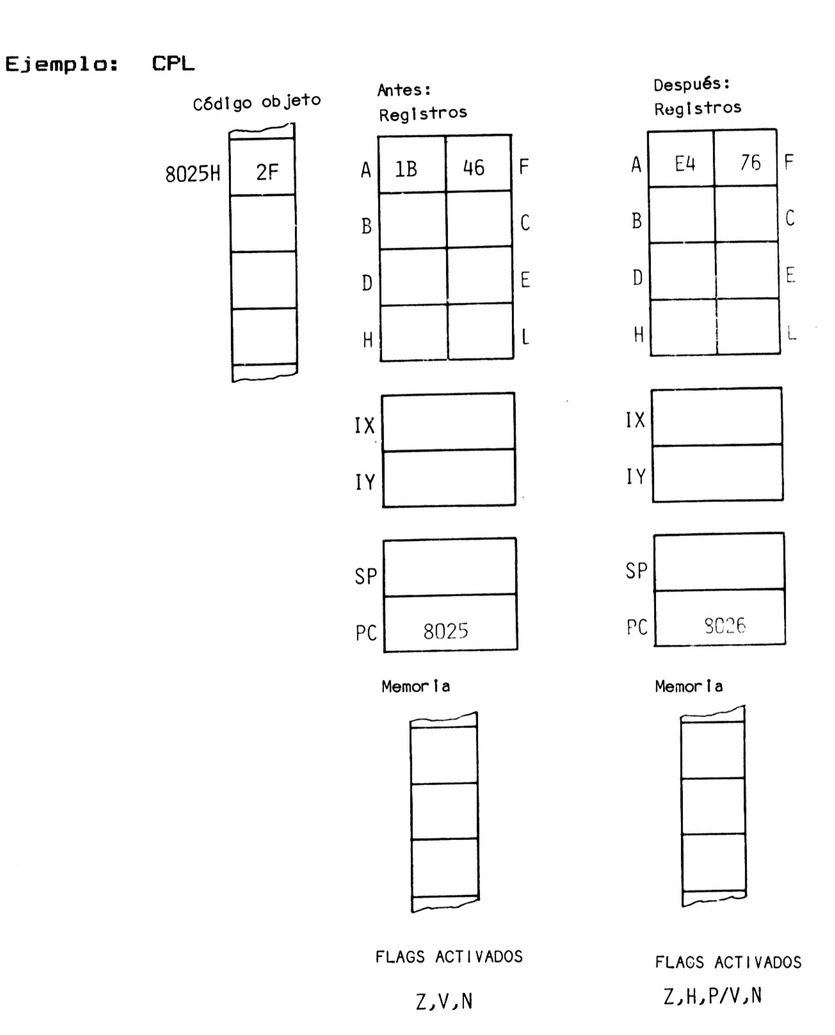
Ejecución:

Tiempo de ejecución	: 1,00 µs
Frecuencia de reloj	: 4,00 MHz
Estados de reloj	: 4
Ciclos de máquina	: 1

Direccionamiento:　　Implícito

Formato:　　0 0 1 0 1 1 1 1　　2FH

Ejemplo: CPL


Código objeto
8025H
2F
Antes:
Registros
A
1B
46
F
B
C
D
E
H
L
IX
IY
SP
PC
8025
Memoria
FLAGS ACTIVADOS
Z,V,N
Después:
Registros
A
E4
76
F
B
C
D
E
H
L
IX
IY
SP
PC
8026
Memoria
FLAGS ACTIVADOS
Z,H,P/V,N

DAA Adaptación decimal del acumulador.

Función: N	C	H	Valor del Nibble sup.	Valor del Nibble inf.	Suma hex. al Acu	C des ejec.
0	0	0	0-9	0-9	00	0
(ADD,	0	0	0-8	A-F	06	0
ADC,	0	1	0-9	0-3	06	0
INC)	0	0	A-F	0-9	60	1
	0	0	9-F	A-F	66	1
	0	1	A-F	0-3	66	1
	1	0	0-2	0-9	60	1
	1	0	0-2	A-F	66	1
	1	1	0-3	0-3	66	1
1	0	0	0-9	0-9	00	0
(SUB,	0	1	0-8	6-F	FA	0
SBC,	1	0	7-F	0-9	A0	1
DEC,	1	1	6-F	6-F	9A	1
NEG)						

Flags:

S	Z	-	H	-	P/V	N	C
X	X		X		X		X

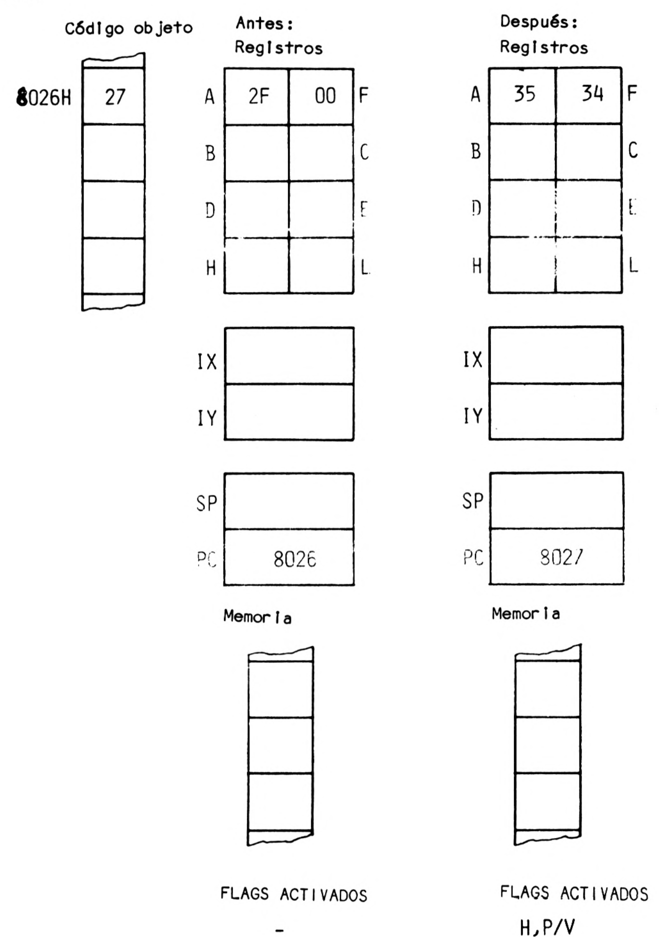
Descripción: Esta instrucción se necesita en la aritmética BCD. Convierte la cifra binaria contenida en el acumulador A en un número de dos posiciones BCD. Mientras tanto, no se efectúa acarreo C (Carry) si la cifra se encuentra por debajo de 100 (decimal). La conversión se realiza mediante la suma de 6H, 66H o bien 60H al acumulador, según el caso.

Ejecución:	Tiempo de ejecución	: 1,00 µs
	Frecuencia de reloj	: 4,00 MHz
	Estados de reloj	: 4
	Ciclos de máquina	: 1

Direccionamiento: Implícito

Formato: 0 0 1 0 0 1 1 1 27H

Ejemplo: DAA



Código objeto
8026H
27
Antes:
Registros
A
2F
00
F
B
C
D
E
H
L
IX
IY
SP
PC
8026
Memoria
FLAGS ACTIVADOS
-
Después:
Registros
A
35
34
F
B
C
D
E
H
L
IX
IY
SP
PC
8027
Memoria
FLAGS ACTIVADOS
H,P/V

DEC m

Decrementa el operando m.

Función: m <——— m - 1

Flags:

S	Z	-	H	-	P/V	N	C
X	X		X		X	1	

Descripción: El operando direccionado a través de n será decrementado.

Ejecución:

m	µs (4 MHz)	Estados T	Ciclos M
r	1,00	4	1
(HL)	2,75	11	3
(IX+d)	5,75	23	6
(IY+d)	5,75	23	6

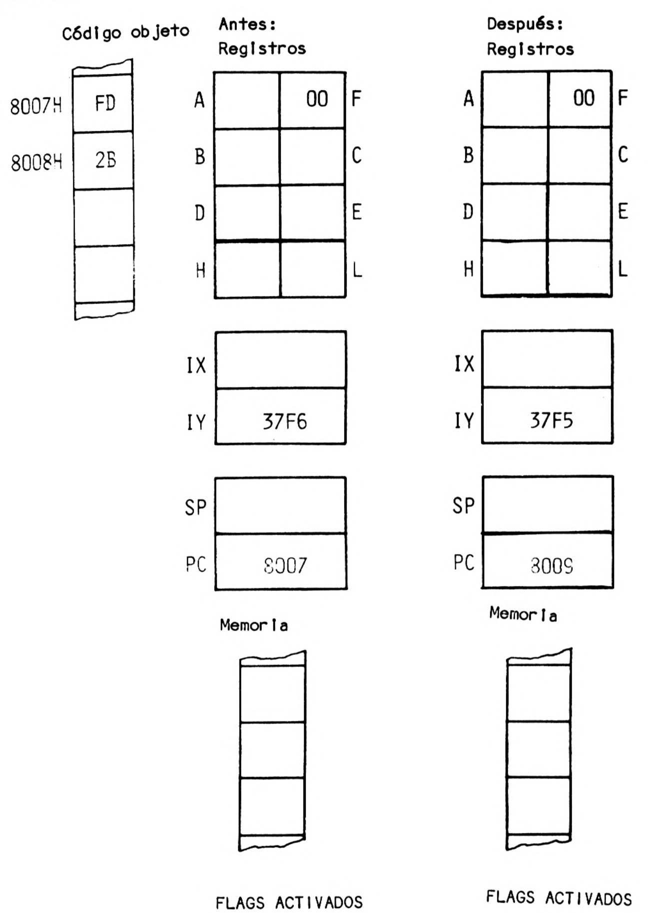
Direccionamiento: r: implícito/ (HL): indirecto/ (IX+d), (IY+d): indexado.

Formato:

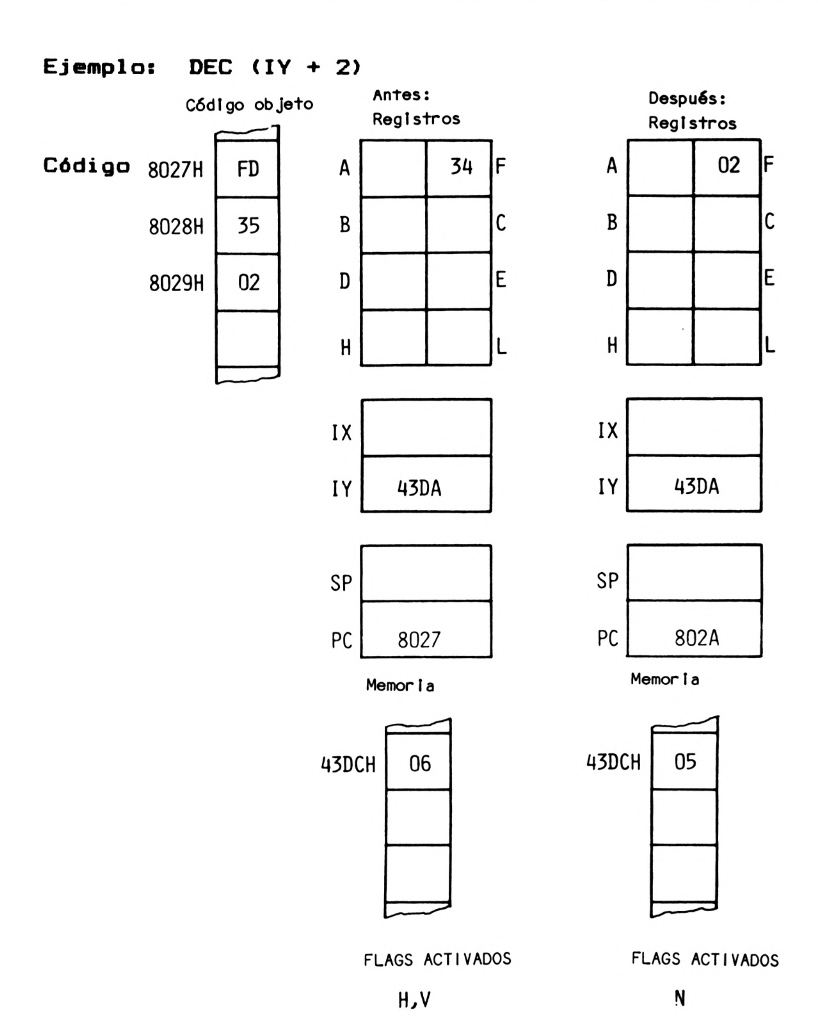
r		0	0-----r-----1					0	1	
(HL)		0	0	1	1	0	1	0	1	35H
(IX+d)	Byte 1)	1	1	0	1	1	1	0	1	DDH
	Byte 2)	0	0	1	1	0	1	0	1	35H
	Byte 3)	-----------d----------Offset								
(IY+d)	Byte 1)	1	1	1	1	1	1	0	1	FDH
	Byte 2)	0	0	1	1	0	1	0	1	35H
	Byte 3)	----------d-----------Offset								

m:puede ser r, (HL), (IX+d), (IY+d)

Ejemplo: DEC IY



Ejemplo: DEC (IY + 2)



FLAGS ACTIVADOS

H,V

FLAGS ACTIVADOS

N

DEC rr Decrementa el par de registros rr.

Función: rr <——— rr - 1

Flags: S Z - H - P/V N C
(ninguna influencia)

Descripción: El par de registros rr se decrementa.

Ejecución:

Tiempo de ejecución	:	1,50 μs
Frecuencia de reloj	:	4,00 MHz
Estados de reloj	:	6
Ciclos de máquina	:	1

Direccionamiento: Implícito

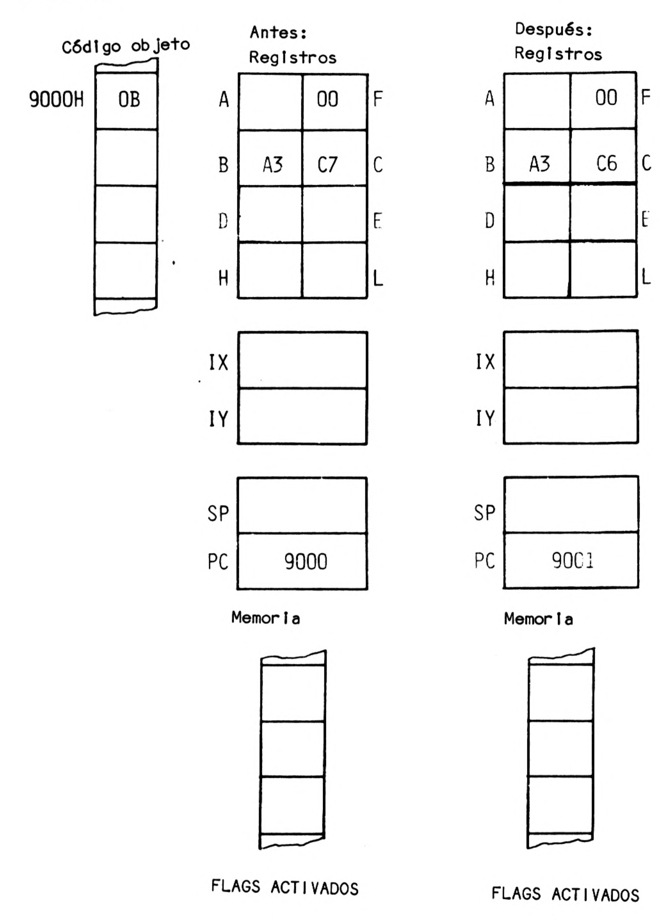
Formato: 0 0 r r 1 0 1 1

rr puede ser:

BC - 00 HL - 10

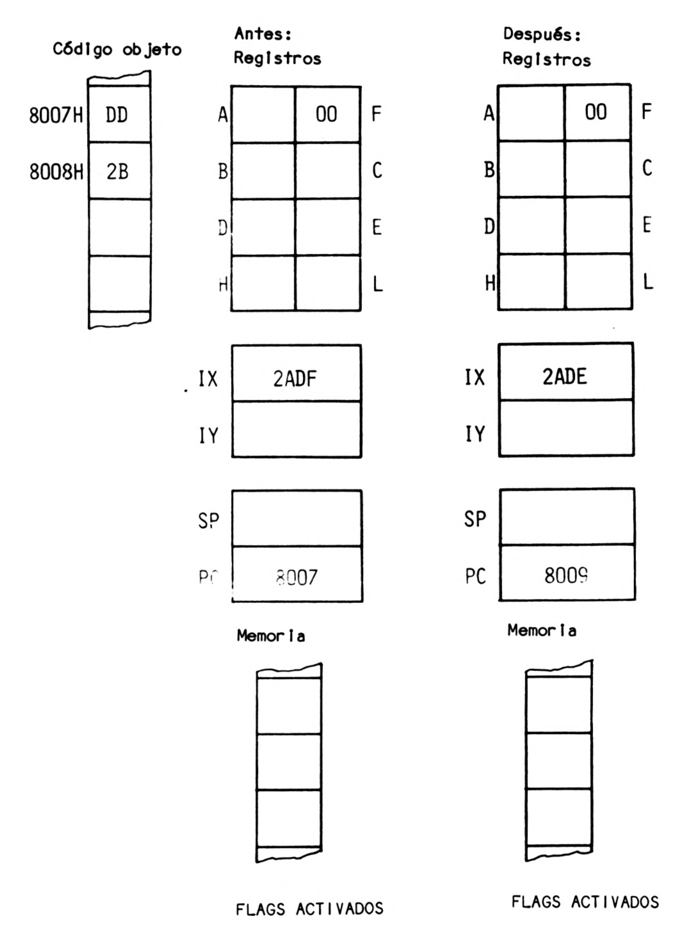
DE - 01 SP - 11

Ejemplo: DEC BC



<u>DEC IX</u>	Decrementa IX
Función:	IX <——— IX - 1
Flags:	S Z - H - P/V N C (ninguna influencia)
Descripción:	El par de registros IX se decrementa.
Ejecución:	Tiempo de ejecución : 2,50 µs Frecuencia de reloj : 4,00 MHz Estados de reloj : 10 Ciclos de máquina : 2
Direccionamiento:	Implícito
Formato:	Byte 1) 1 1 0 1 1 1 0 1 DDH Byte 2) 0 0 1 0 1 0 1 1 2BH

Ejemplo: DEC IX



DEC IY — **Decrementa el IY.**

Función: IY <——— IY - 1

Flags: S Z - H - P/V N C
(ninguna influencia)

Descripción: El par de registros IY se decrementa.

Ejecución:

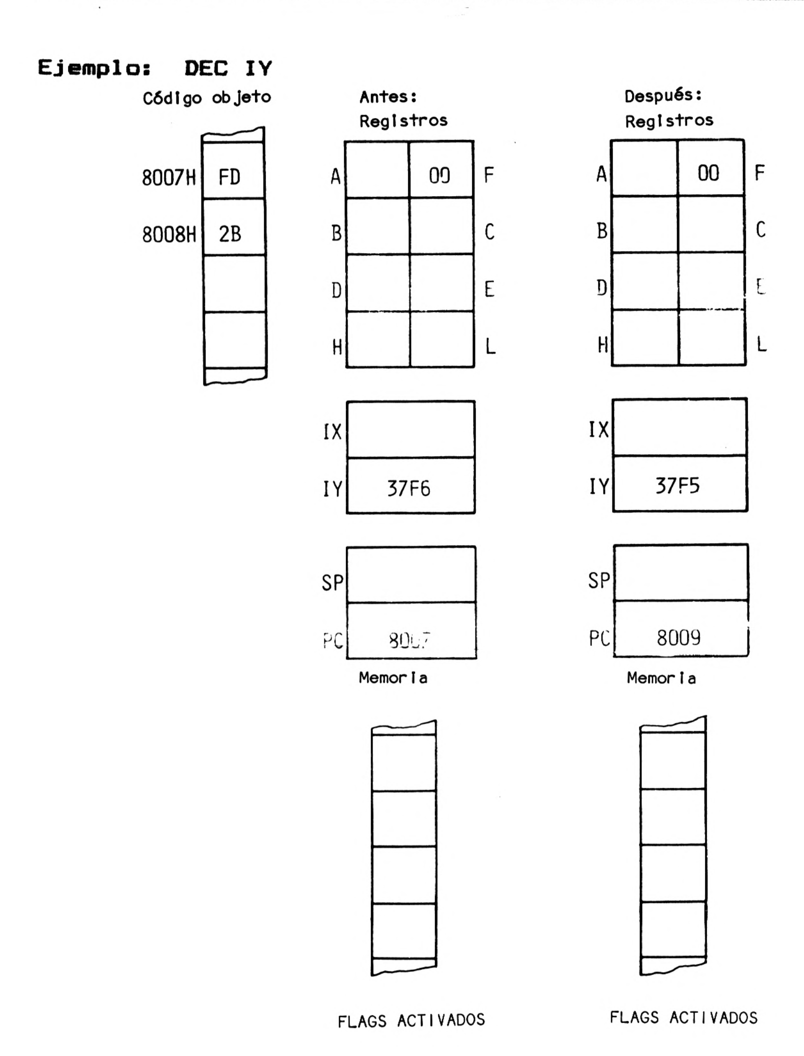
Tiempo de ejecución	: 2,50 µs
Frecuencia de reloj	: 4,00 MHz
Estados de reloj	: 10
Ciclos de máquina	: 2

Direccionamiento: Implícito

Formato:

Byte 1)	1	1	1	1	1	1	0	1	FDH
Byte 2)	0	0	1	0	1	0	1	1	2BH

Ejemplo: DEC IY



FLAGS ACTIVADOS FLAGS ACTIVADOS

DI — Bloquea el interrupt (disable interrupt).

Función: IFF -- 0

Flags: S Z - H - P/V N C
(ninguna influencia)

Descripción: Con esta instrucción se bloquea la entrada INT, permaneciendo desde este momento bajo máscara. Internamente en el procesador ésto se produce desactivando (valor 0) el interrupt flip-flop (IFF1 e IFF2). La entrada INT bloqueada puede liberarse a través de la instrucción "EI" (Enable Interrupt); es decir, que una señal de cero en esta entrada podrá provocar nuevamente interrupciones.

Ejecución:

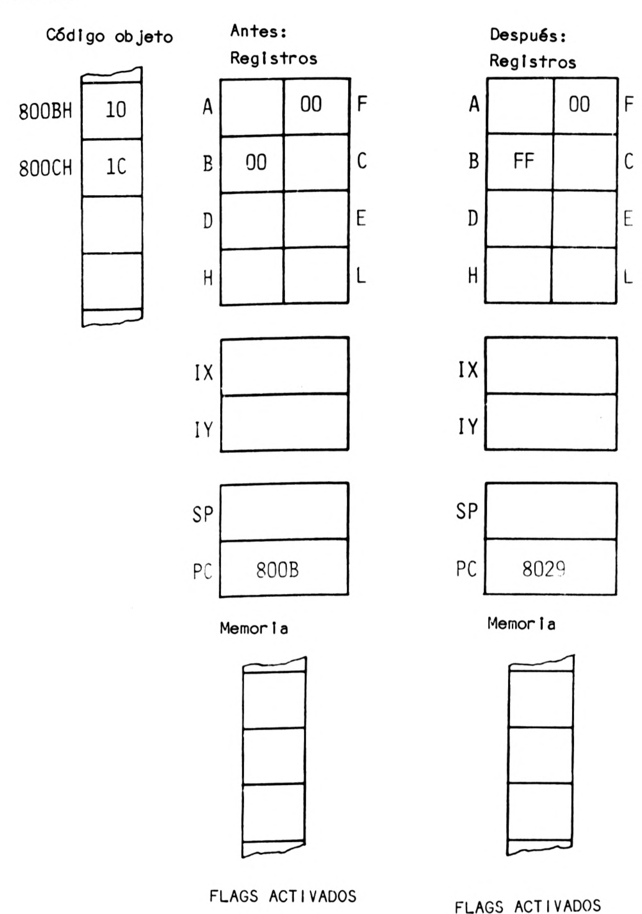
Tiempo de ejecución	:	1,00 µs
Frecuencia de reloj	:	4,00 MHz
Estados de reloj	:	4
Ciclos de máquina	:	1

Direccionamiento: Implícito

Formato: 1 1 1 1 0 0 1 1 F3H

DJNZ e	Decrementa B y efectúa un salto distinto de cero.
Función:	B -- B - 1;en caso de B≠0: PC--PC+e
Flags:	S Z - H - P/V N C (ninguna influencia)
Descripción:	Esta instrucción se utiliza muy a menudo en bucles que no necesiten más de 256 ciclos y en los que se pueda trabajar en un rango de bifurcaciones efectivo de + 129 hasta - 126. Después de decrementar el registro B se verificará su contenido comparando con cero. Unicamente cuando B no sea igual a cero, se bifurcará a la dirección que resultante de sumar el contenido del contador de programa después de la ejecución de la instrucción (PC + 2) y desplazamiento e direccional con la técnica complemento binario.
Ejecución:	B ≠ 0: 3 ciclos M; 13 estados T; 3,25 μs 4,00 MHz B = 0: 2 ciclos M; 8 estados T; 2,00 μs 4,00 MHz
Direccionamiento:	Inmediato
Formato:	Byte 1) 0 0 0 1 0 0 0 0 10H Byte 2) ----------e-2---------Offset

Ejemplo: DJNZ 28



EI Libera interrupts.

Función: IFF -- 1

Flags: S Z - H - P/V N C
(ninguna influencia)

Descripción: Con esta instrucción se desbloquea la entrada INT de interrupts. Internamente, en el procesador, se activan (valor 1) Los interrupts flip-flop IFF1 e IFF2 (véase también "DI"). El desbloqueo es efectivo a partir de la instrucción siguiente al EI.

Ejecución:

Tiempo de ejecución	:	1,00 µs
Frecuencia de reloj	:	4,00 MHz
Estados de reloj	:	4
Ciclos de máquina	:	1

Direccionamiento: Implícito

Formato: 1 1 1 1 1 0 1 1 FBH

EX (SP),HL — Intercambia HL con el contenido de los dos últimos elementos la pila.

Función: (SP) <——— L; (SP + 1) -- H

Flags: S Z - H - P/V N C
(ninguna influencia)

Descripción: El contenido del par de registros HL se intercambia con las dos entradas de la pila indicadas por el apuntador de pilas (SP). El primer elemento señalado por el apuntador de pila se carga en el registro L. El apuntador de pila se incrementa apuntando entonces al segundo elemento, que se carga en el registro H.

Ejecución:

Tiempo de ejecución	:	4,75 µs
Frecuencia de reloj	:	4,00 MHz
Estados de reloj	:	19
Ciclos de máquina	:	5

Direccionamiento: Indirecto

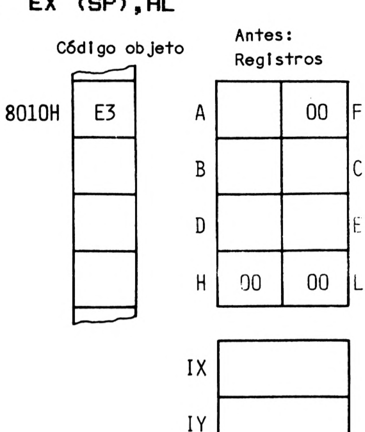
Formato: 1 1 1 0 0 0 1 1 E3H

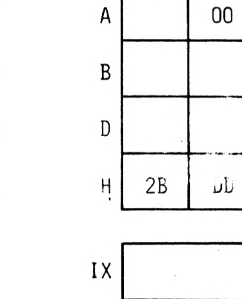
Ejemplo: EX (SP),HL

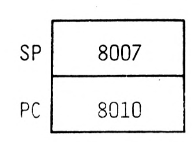
Código objeto

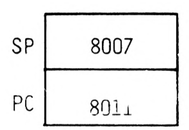
Antes:
Registros

Después:
Registros

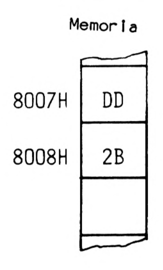




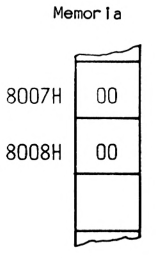




Memoria



Memoria



FLAGS ACTIVADOS

-

FLAGS ACTIVADOS

-

EX (SP),IX — Intercambia IX con el contenido de los dos últimos elementos de la pila.

Función: (SP) <——— IX_{abajo}; (SP+1)--IX_{arriba}

Flags: S Z - H - P/V N C
(ninguna influencia)

Descripción: El contenido del par de registros IX se intercambia con las 2 entradas de la pila (Stack) indicadas por el apuntador de pila (Stack-pointer). El primer elemento señalado por el apuntador de pila, se carga en el registro X.El apuntador de pila se incrementa señalando entonces al segundo elemento, que se carga ahora en el registro I.

Ejecución:

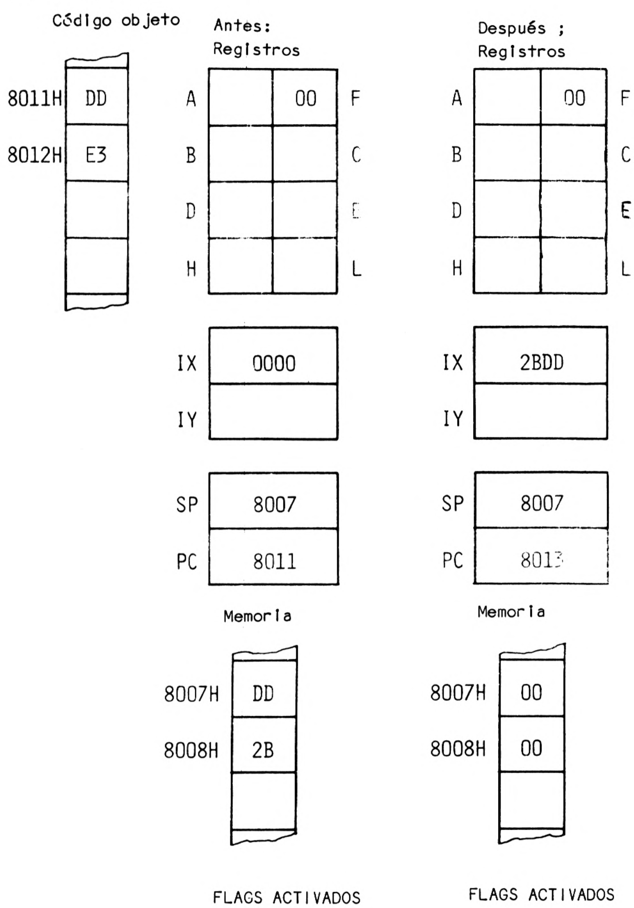
Tiempo de ejecución	:	5,75 µs
Frecuencia de reloj	:	4,00 MHz
Estados de reloj	:	23
Ciclos de máquina	:	6

Direccionamiento: Indirecto

Formato:

Byte 1)	1	1	0	1	1	1	0	1	DDH
Byte 2)	1	1	1	0	0	0	1	1	E3H

Ejemplo: EX (SP),IX



EX (SP),IY — Intercambia IY con el contenido de los dos últimos elementos de la pila.

Función: (SP) <——— IY_{abajo}; (SP+1)--IY_{arriba}

Flags: S Z - H - P/V N C
(ninguna influencia)

Descripción: El contenido del par de registros IY se intercambia con las dos entradas de la pila (Stack) indicadas por el apuntador de pila (Stack-pointer). El primer elemento señalado por este apuntador de pila, se carga en el registro Y. A continuación se incrementa el apuntador indicando ahora al segundo elemento, que se carga entonces en el registro I.

Ejecución:

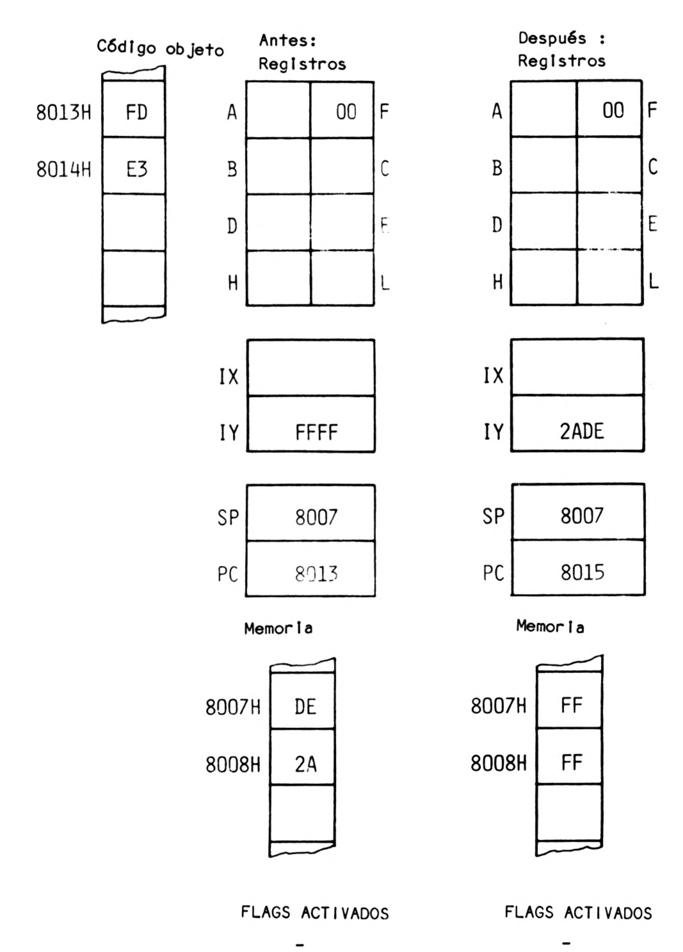
Tiempo de ejecución	:	5,75 µs
Frecuencia de reloj	:	4,00 MHz
Estados de reloj	:	23
Ciclos de máquina	:	6

Direccionamiento: Indirecto

Formato:

Byte 1)	1	1	1	1	1	1	0	1	FDH
Byte 2)	1	1	1	0	0	0	1	1	E3H

Ejemplo: EX (SP),IY



EX AF, AF' — **Intercambia el acumulador y los flags con los registros complementarios.**

Función: AF <——— AF'

Flags: S Z - H - P/V N C
(ninguna influencia)

Descripción: El microprocesador Z80 posee dos juegos de registros universales idénticos, cada uno de 8 registros. Con esta instrucción se intercambiarán los contenidos de los registros del par de registros AF con su par de registros complementarios AF'.

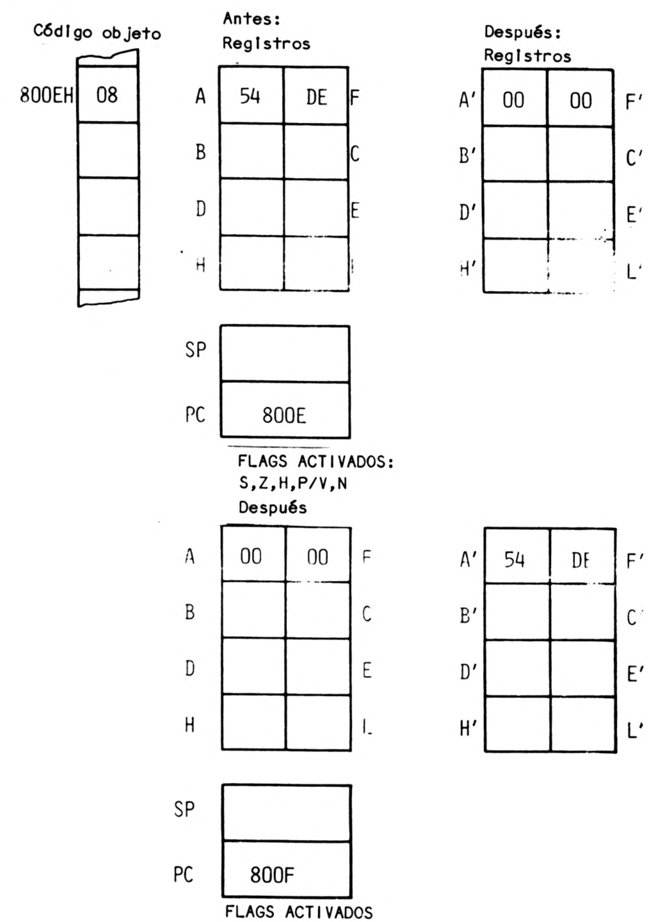
Ejecución:

Tiempo de ejecución	: 1,00 µs
Frecuencia de reloj	: 4,00 MHz
Estados de reloj	: 4
Ciclos de máquina	: 1

Direccionamiento: Implícito

Formato: 0 0 0 0 1 0 0 0 08H

Ejemplo: EX AF,AF'



EX DE,HL Intercambia los registro HL y DE.

Función: DE <——— HL

Flags: S Z - H - P/V N C
(ninguna influencia)

Descripción: Los contenidos del par de registros HL y DE se intercambiarán, es decir que DE contendrá, después de la ejecución de la instrucción, el literal que antes estaba en el HL; lo mismo ocurrirá con HL.

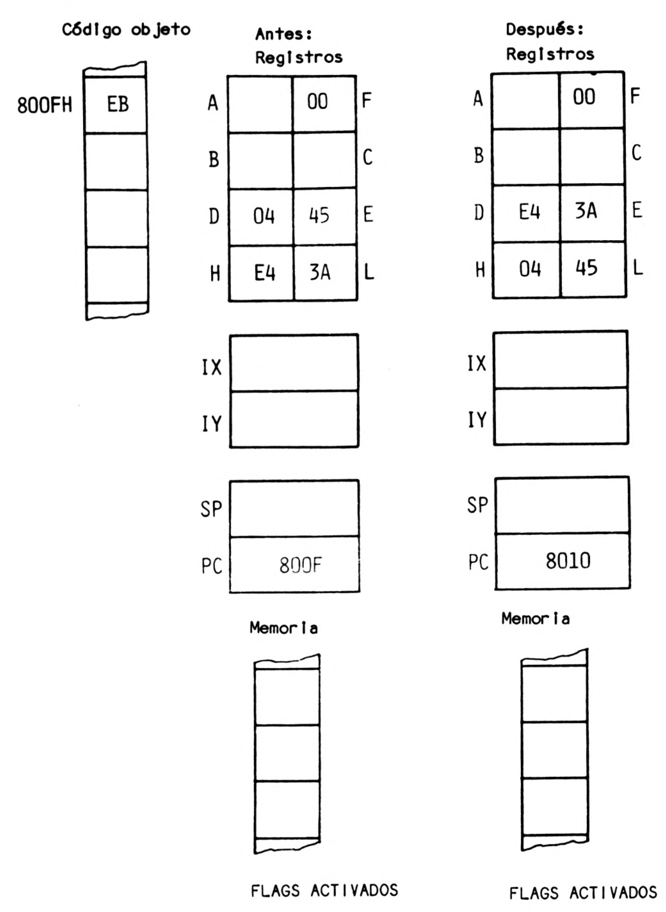
Ejecución:

Tiempo de ejecución	:	1,00 µs
Frecuencia de reloj	:	4,00 MHz
Estados de reloj	:	4
Ciclos de máquina	:	1

Direccionamiento: Implícito

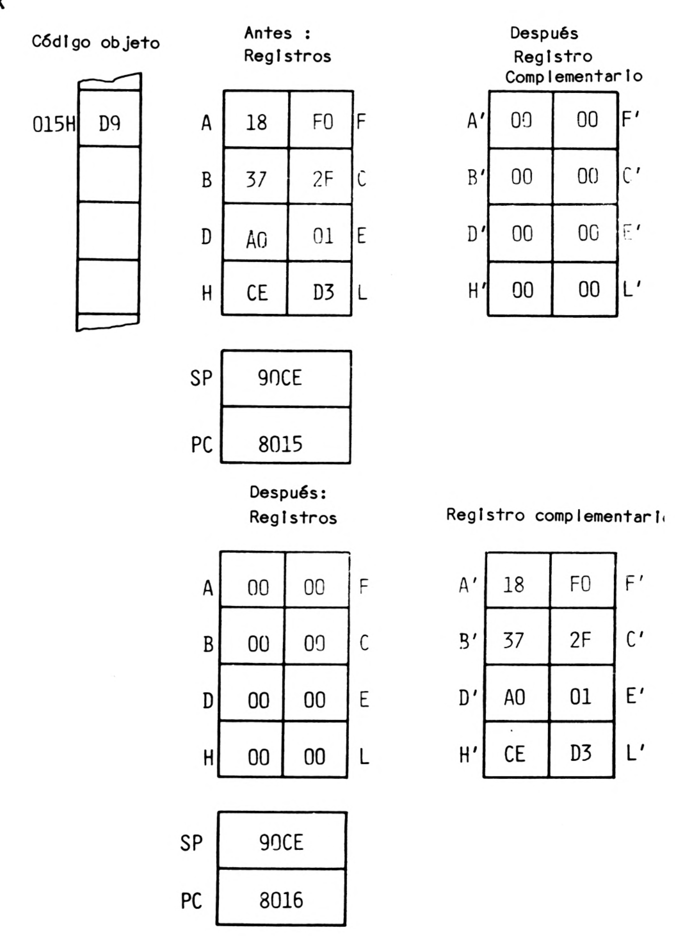
Formato: 1 1 1 0 1 0 1 1 EBH

Ejemplo: EX DE,HL



EXX	Intercambia con los registros complementarios.
Función:	BC <——— BC; DE -- DE; HL -- HL
Flags:	S Z - H - P/V N C (ninguna influencia)
Descripción:	El contenido de los 8 registros universales se intercambiará con el contenido del juego de registros complementarios.
Ejecución:	Tiempo de ejecución : 1,00 µs Frecuencia de reloj : 4,00 MHz Estados de reloj : 4 Ciclos de máquina : 1
Direccionamiento:	Implícito
Formato:	1 1 0 1 1 0 0 1 D9H

Ejemplo: EXX


Código objeto
015H
D9
Antes :
Registros
A
18
F0
F
B
37
2F
C
D
A0
01
E
H
CE
D3
L
Después
Registro
Complementario
A'
00
00
F'
B'
00
00
C'
D'
00
00
E'
H'
00
00
L'
SP
90CE
PC
8015
Después:
Registros
A
00
00
F
B
00
00
C
D
00
00
E
H
00
00
L
Registro complementario
A'
18
F0
F'
B'
37
2F
C'
D'
A0
01
E'
H'
CE
D3
L'
SP
90CE
PC
8016

<u>HALT</u>	**Parar la CPU.**
Función:	CPU
Flags:	S Z - H - P/V N C (ninguna influencia)
Descripción:	Esta instrucción se aplicará, cuando un programa tenga que ser interrumpido en una posición determinada hasta que intervenga un interrupt o bien un reset. Mientras el procesador esté esperando seguirá ejecutando instrucciones NOP, que en realidad no efectúan nada pero que son importantes para la emisión de direcciones de refresco destinadas a memorias dinámicas eventualmente conectadas.
Ejecución:	Tiempo de ejecución : 1,00 µs Frecuencia de reloj : 4,00 MHz + NOP's no definidos Estados de reloj : 4 Ciclos de máquina : 1
Direccionamiento:	Implícito
Formato:	0 1 1 1 0 1 1 0 76H

IM 0	**Activa modalidad de interrupt 0.**
Función:	Control interno de interrupt.
Flags:	S Z - H - P/V N C (ninguna influencia)
Descripción:	Esta modalidad también está disponible en el microprocesador 80 80. Ha sido tomada del Z80 con el fin de poder garantizar una "compatibilidad ascendente". El componente a ser interrumpido envía en primer lugar un byte al bus de datos. Este byte purde ser una de las 8 posibles instrucciones RST o bien una instrucción CALL (CDH). En el caso de ser una instrucción CALL, el Z80 lee dos bytes más del bus de datos y los trata como direcciones de bifurcación para la correspondiente rutina de manejo del interrupt. Si el byte es un comando RST, resulta una bifurcación a la dirección determinada de forma implícita por el propio comando RST en el registro de página cero (Zero Page).
Ejecución:	Tiempo de ejecución : 2,00 µs Frecuencia de reloj : 4,00 MHz Estados de reloj : 8 Ciclos de máquina : 2
Direccionamiento:	Implícito
Formato:	Byte 1) 1 1 1 0 1 1 0 1 EDH Byte 2) 0 1 0 0 0 1 1 0 46H

IM 1 — Activa la modalidad 1 de interrupt.

Función:	Control interno de interrupt.
Flags:	S Z - H - P/V N C (ninguna influencia)
Descripción:	Cuando esta modalidad se ha inicializado, en el caso de un interrupt se efectuará automáticamente una bifurcación a la dirección 38H, sin leer ningún otro byte. Aquí empezará la rutina correspondiente de interrupt.
Ejecución:	Tiempo de ejecución : 2,00 µs Frecuencia de reloj : 4,00 MHz Estados de reloj : 8 Ciclos de máquina : 2
Direccionamiento:	Implícito
Formato:	Byte 1) 1 1 1 0 1 1 0 1 EDH Byte 2) 0 1 0 1 0 1 1 0 56H

IM 2 — Activa la modalidad 2 de interrupt.

Función: Control interno de interrupt.

Flags: S Z - H - P/V N C
(ninguna influencia)

Descripción: Esta modalidad es muy elegante y variada. En relación con los componentes periféricos del Z80 podrán manipularse hasta 128 rutinas de interrupt sin tener que aplicar componentes de control de prioridad (PIC). La modalidad trabaja con vectores direccionales, cuya mitad superior se obtiene del registro I y su mitad inferior se envía al bus de datos desde el componente que provoca el interrupt. En conjunto resultará un vector de 16 bits. Mediante la carga correspondiente del registro I, dichos vectores podrán indicar cualquier posición en el área de memoria de 64 K-bytes. Estos vectores apuntarán sobre la llamada "tabla de bifurcaciones" confeccionda previamente en memoria. Las direcciones de bifurcación de dicha tabla señalarán las rutinas correspondientes de interrupt. De este modo se tendrá la elegante posibilidad de intercalar la tabla de bifurcaciones en cualquier campo de memoria y por otro lado se dispondrá de la libertad de disponer las rutinas de interrupción, independientemente unas de otras, en las posiciones adecuadas del área total de memoria.

Ejecución:	Tiempo de ejecución	: 2,00 µs
	Frecuencia de reloj	: 4,00 MHz
	Estados de reloj	: 8
	Ciclos de máquina	: 2

Direccionamiento: Implícito

Formato:										
	Byte 1)	1	1	1	0	1	1	0	1	EDH
	Byte 2)	0	1	0	1	1	1	1	0	5EH

IN A, (N) Carga el acumulador del port N.

Función: A <——— (N)

Flags: S Z - H - P/V N C
(ninguna influencia)

Descripción: El acumulador A se cargará mediante el operando (N) direccionado a través del literal N. La N suministrará la dirección en página cero. En las líneas direccionales A8.....A15 aparecerá el contenido del registro A.

Ejecución:

Tiempo de ejecución	:	2,75 µs
Frecuencia de reloj	:	4,00 MHz
Estados de reloj	:	11
Ciclos de máquina	:	3

Direccionamiento: Directo externo.

Formato:

Byte 1) 1 1 0 1 1 0 1 1 DBH

Byte 2) -----------N----------Direc. Port

Formato:	Byte 1)	1	1	1	0	1	1	0	1	EDH
	Byte 2)	0	1	---r------			0	0	0	

r puede ser:

A - 111	E - 011
B - 000	H - 100
C - 001	L - 101
D - 010	

Ejemplo: IN A,(1)

Antes:

Código objeto

80018H	DB
80019H	01

Registros

A	00	20	F
B			C
D			E
H			L

IX	
IY	

SP	
PC	8018

PORT

01H	20

Flags activados:

Después:

Registros

A	20	20	F
B			C
D			E
H			L

IX	
IY	

SP	
PC	801A

PORT

01H	20

Flags activados:

IN r,(C)

Carga el registro r del port (C).

Función: r <——— (C)

Flags:

S	Z	-	H	-	P/V	N	C
X	X		X		X	0	

Es importante tener en cuenta que IN A,(N) no tiene ninguna influencia sobre los flags; sin embargo sí la tiene IN r,(C).

Descripción: El registro r se cargará mediante el operando (C) direccionado externamente a través del registro C. La C suministrará la dirección a la página cero. Sobre las líneas direccionales A8.....A15 aparecerá el contenido del registro B.

Ejecución:

Tiempo de ejecución	: 3,00 µs
Frecuencia de reloj	: 4,00 MHz
Estados de reloj	: 12
Ciclos de máquina	: 3

Direccionamiento: Indirecto externo

Ejemplo: IN D(C)

Código objeto		Antes: Registros				Después: Registros			
8016H	ED	A		00	F	A		20	F
8017H	50	B		01	C	B		01	C
		D	00		E	D	20		E
		H			L	H			L

Antes		Después	
IX		IX	
IY		IY	
SP		SP	
PC	8016	PC	8018

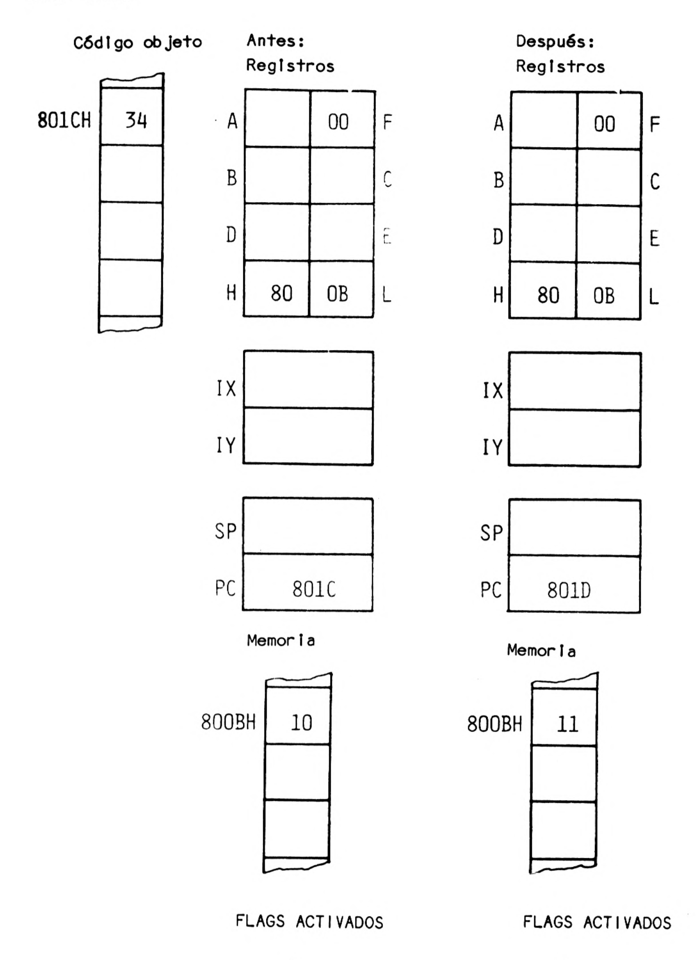
Antes: PORT		Después: PORT	
01H	20	01H	20

Flags activados:

Flags activados:

INC (HL)	Incrementa la posición de memoria (HL) indirectamente direccionada.
Función:	(HL) <——— (HL) + 1
Flags:	S Z - H - P/V N C X X X X 0
Descripción:	Se incrementa el operando (HL) indirectamente direccionado a través del par de registros HL.
Ejecución:	Tiempo de ejecución : 2,75 µs Frecuencia de reloj : 4,00 MHz Estados de reloj : 11 Ciclos de máquina : 3
Direccionamiento:	Indirecto
Formato:	0 0 1 1 0 1 0 0 34H

Ejemplo: INC (HL)



FLAGS ACTIVADOS

-

FLAGS ACTIVADOS

-

INC IX

Incrementa IX.

Función: IX <——— IX + 1

Flags: S Z - H - P/V N C
(ninguna influencia)

Descripción: Se incrementa el registro de índices IX.

Ejecución:

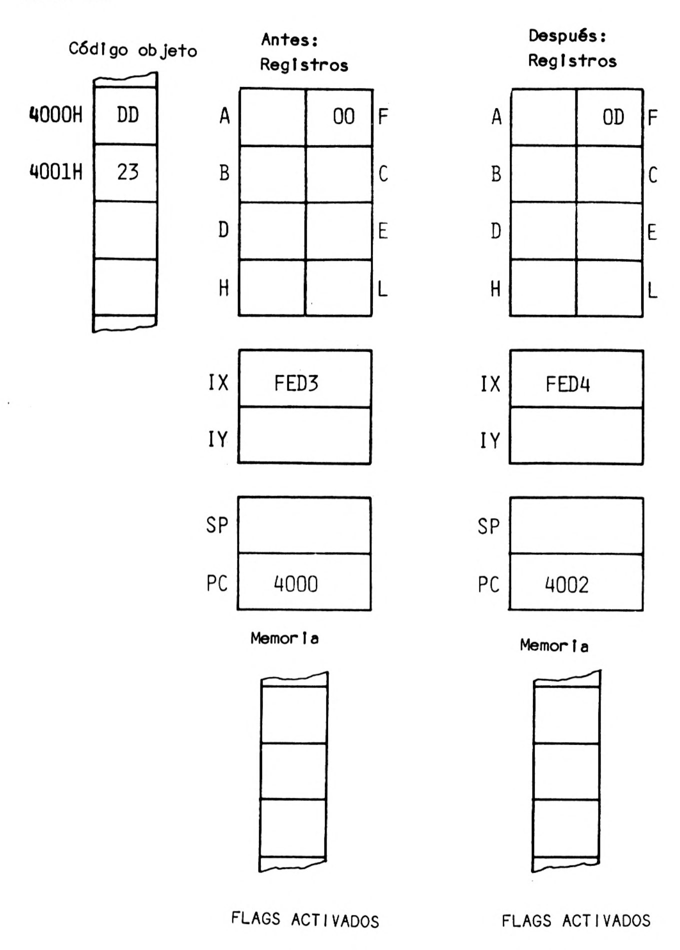
Tiempo de ejecución	: 2,50 µs
Frecuencia de reloj	: 4,00 MHz
Estados de reloj	: 10
Ciclos de máquina	: 2

Direccionamiento: Implícito

Formato:

Byte 1)	1	1	0	1	1	1	0	1	DDH
Byte 2)	0	0	1	0	0	0	1	1	23H

Ejemplo: INC IX



INC (IX + d) Incrementa la posición de memoria (IX + d) direccionada indexadamente.

Función: (IX + d) <——— (IX + d) + 1

Flags:

S	Z	-	H	-	P/V	N	C
X	X		X		X	0	

Descripción: Se incrementa el operando direccionado indexadamente a través del registro de índices IX, además del desplazamiento d.

Ejecución:

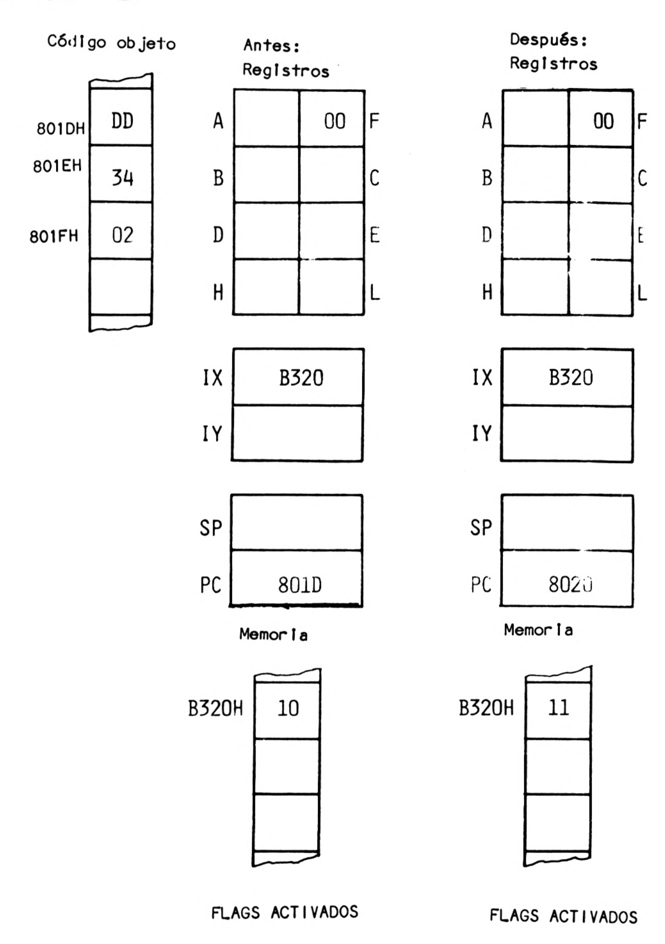
Tiempo de ejecución : 5,75 µs
Frecuencia de reloj : 4,00 MHz
Estados de reloj : 23
Ciclos de máquina : 6

Direccionamiento: Indexado

Formato:

Byte 1)	1	1	0	1	1	1	0	1	DDH
Byte 2)	0	0	1	1	0	1	0	0	34H
Byte 3)	----------d----------								Offset

Ejemplo: INC (IX + 2)



FLAGS ACTIVADOS

-

FLAGS ACTIVADOS

-

<u>INC IY</u> Incrementa el IY.

Función: IY <——— IY + 1

Flags: S Z - H - P/V N C
(ninguna influencia)

Descripción: Se incrementa el registro de índices IY.

Ejecución:

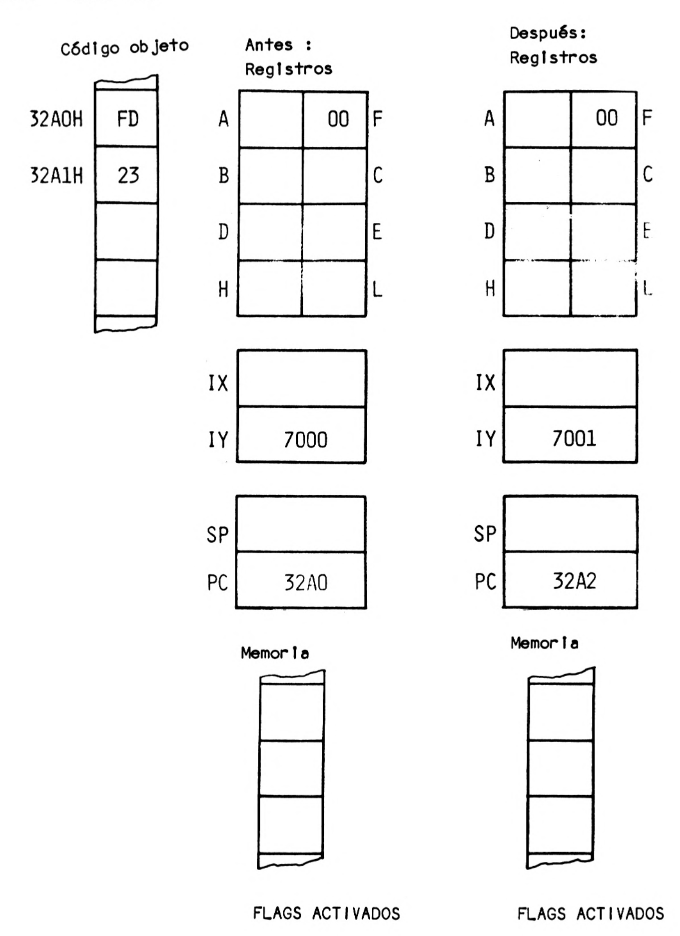
Tiempo de ejecución	:	2,50 µs
Frecuencia de reloj	:	4,00 MHz
Estados de reloj	:	10
Ciclos de máquina	:	2

Direccionamiento: Implícito

Formato:

Byte 1)	1	1	1	1	1	1	0	1	FDH
Byte 2)	0	0	1	0	0	0	1	1	23H

Ejemplo: INC IY



INC (IY + d) Incrementa la posición de memoria (IY + d) direccionada indexadamente.

Función: (IY + d) <——— (IY + d) + 1

Flags:

S	Z	-	H	-	P/V	N	C
X	X		X		X	0	

Descripción: Se incrementa el operando direccionado indexadamente a través del registro de índices IY, y del desplazamiento d.

Ejecución:

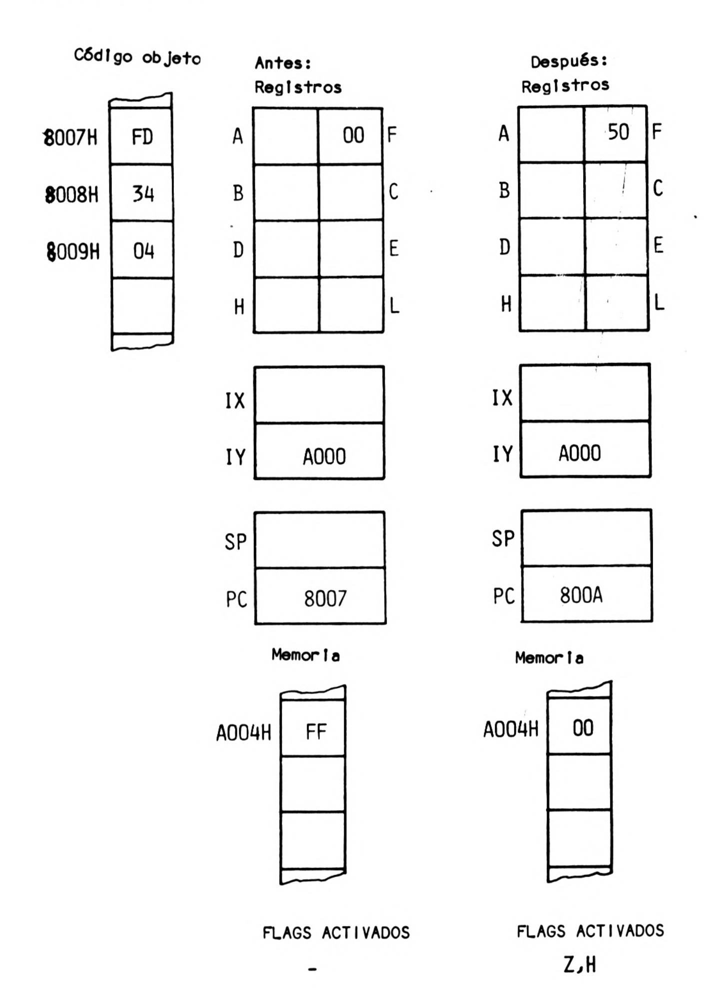
Tiempo de ejecución	: 5,75 µs
Frecuencia de reloj	: 4,00 MHz
Estados de reloj	: 23
Ciclos de máquina	: 6

Direccionamiento: Indexado

Formato:

Byte 1)	1	1	1	1	1	1	0	1	FDH
Byte 2)	0	0	1	1	0	1	0	0	34H
Byte 3)	----------d----------Offset								

Ejemplo: INC (IY + 4)



INC r

Incrementa el registro r.

Función: R <——— r + 1

Flags:

S	Z	-	H	-	P/V	N	C
X	X		X		X	O	

Descripción: El registro r será incrementado.

Ejecución:

Tiempo de ejecución	: 1,00 µs
Frecuencia de reloj	: 4,00 MHz
Estados de reloj	: 4
Ciclos de máquina	: 1

Direccionamiento: Implícito

Formato:

0 0-----r-----1 0 0

r puede ser:

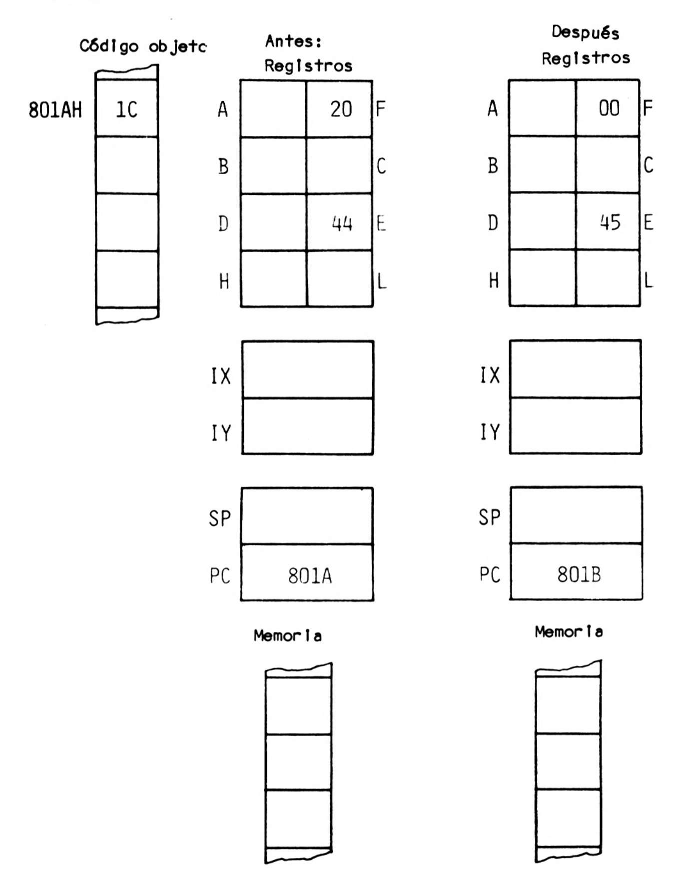
A - 111 E - 011

B - 000 H - 100

C - 001 L - 101

D - 010

Ejemplo: INC E



FLAGS ACTIVADOS

-

FLAGS ACTIVADOS

-

INC rr — Incrementa el par de registros rr.

Función: rr <——— rr + 1

Flags: S Z - H - P/V N C
(ninguna influencia)

Descripción: El par de registros rr será incrementado.

Ejecución:

Tiempo de ejecución	:	1,50 μs
Frecuencia de reloj	:	4,00 MHz
Estados de reloj	:	6
Ciclos de máquina	:	1

Direccionamiento: Implícito

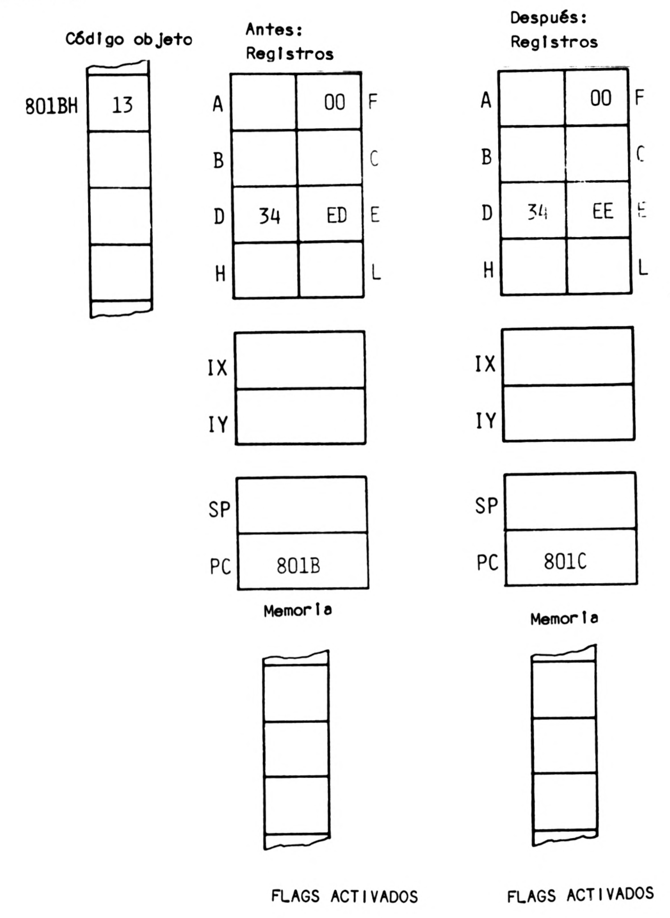
Formato:

0 0 r r 0 0 1 1

rr puede ser:

BD - 00	HL - 10
DE - 01	SP - 11

Ejemplo: INC DE



FLAGS ACTIVADOS

-

FLAGS ACTIVADOS

-

IND **Entrada con decremento.**

Función: (HL) <— (C); B <— B - 1; HL<—HL -1

Flags:

S	Z	-	H	-	P/V	N	C
?	X		?		?	1	Activado si B=0 después de la ejec., de lo contr. desactivado

Descripción: La posición de memoria (HL) direccionada indirectamente a través del par de registros HL será cargada mediante el operando (C) direccionado externamente a través del registro C. A continuación B HL serán decrementadas.

Ejecución:

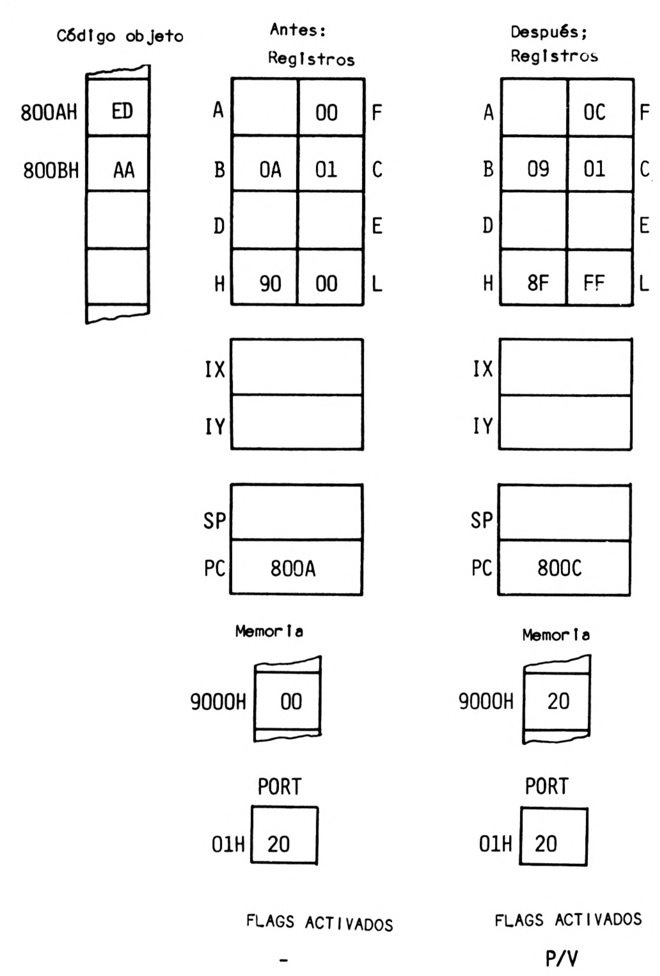
Tiempo de ejecución : 4,00 µs
Frecuencia de reloj : 4,00 MHz
Estados de reloj : 16
Ciclos de máquina : 4

Direccionamiento: Externo

Formato:

Byte 1)	1	1	1	0	1	1	0	1	EDH
Byte 2)	1	0	1	0	1	0	1	0	AAH

Ejemplo: **IND**



Código objeto
800AH
ED
800BH
AA
Antes:
Registros
A
00
F
B
0A
01
C
D
E
H
90
00
L
IX
IY
SP
PC
800A
Memoria
9000H
00
PORT
01H
20
FLAGS ACTIVADOS
-
Después;
Registros
A
0C
F
B
09
01
C
D
E
H
8F
FF
L
IX
IY
SP
PC
800C
Memoria
9000H
20
PORT
01H
20
FLAGS ACTIVADOS
P/V

INDR — **Entrada de bloques con decremento.**

Función: (HL)<—(C); B<—B-1; HL<—HL-1
Repetición hasta B = 0

Flags:

S	Z	-	H	-	P/V	N	C
?	1		?		?	1	

Descripción: La posición de memoria (HL) direccionada indirectamente a través del par de registros HL será cargada mediante el operando (C) direccionado externamente a través del registro C. A continuación B y HL se decrementan. El contenido del registro B se compara con cero. Unicamente cuando B sea diferente a cero, el contador de programas se decrementa en 2 y de este modo volverá a ejecutarse la instrucción.

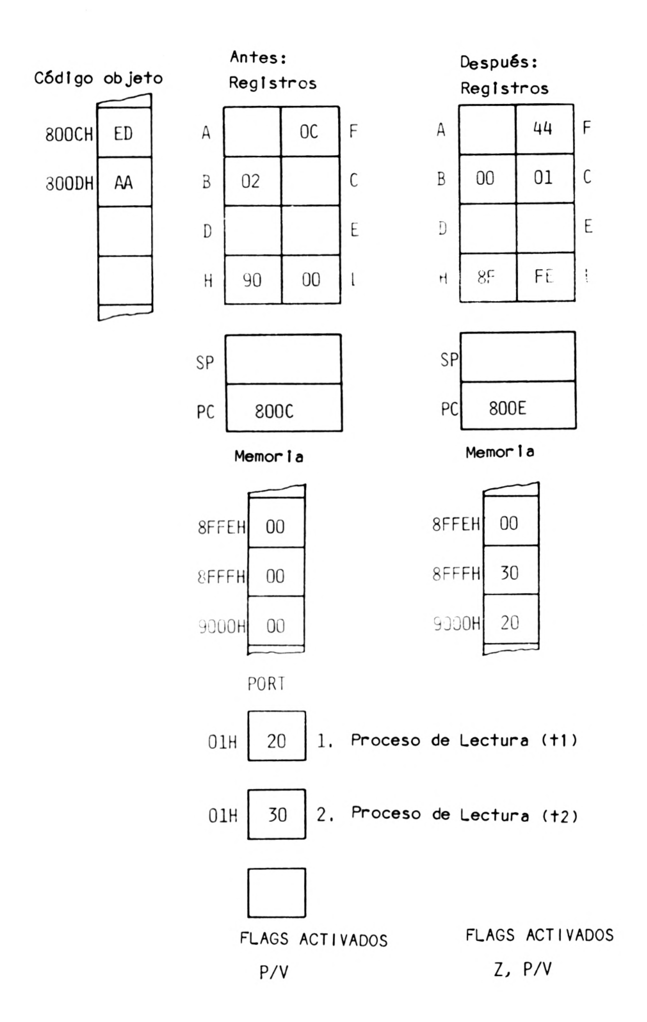
Ejecución: B = 0: 4 ciclos M; 16 estados T; 4 µs 4,00 MHz
B = 0: 5 ciclos M; 21 estados T; 5,25 µs 4,00 MHz.

Direccionamiento: Externo

Formato:

Byte 1)	1	1	1	0	1	1	0	1	EDH
Byte 2)	1	0	1	1	1	0	1	0	BAH

Ejemplo: INDR


Código objeto
800CH
ED
800DH
AA
Antes:
Registros
A
0C
F
B
02
C
D
E
H
90
00
L
SP
PC
800C
Memoria
8FFEH
00
8FFFH
00
9000H
00
PORT
01H
20
1. Proceso de Lectura (t1)
01H
30
2. Proceso de Lectura (t2)
FLAGS ACTIVADOS
P/V
Después:
Registros
A
44
F
B
00
01
C
D
E
H
8F
FE
L
SP
PC
800E
Memoria
8FFEH
00
8FFFH
30
9000H
20
FLAGS ACTIVADOS
Z, P/V

INI — Entrada con incremento.

Función: (HL)<— (C); B<— B-1; HL<—HL +1

Flags:

S	Z	-	H	-	P/V	N	C
?	X		?		?	1	

Z se activa si después de la ejecución B = 0, en caso contrario se desactiva.

Descripción: La posición de memoria (HL) direccionada indirectamente a través del par de registros HL será cargada mediante el operando (C) direccionado externamente a través del registro C. A continuación se decrementa B y se incrementa HL.

Ejecución:

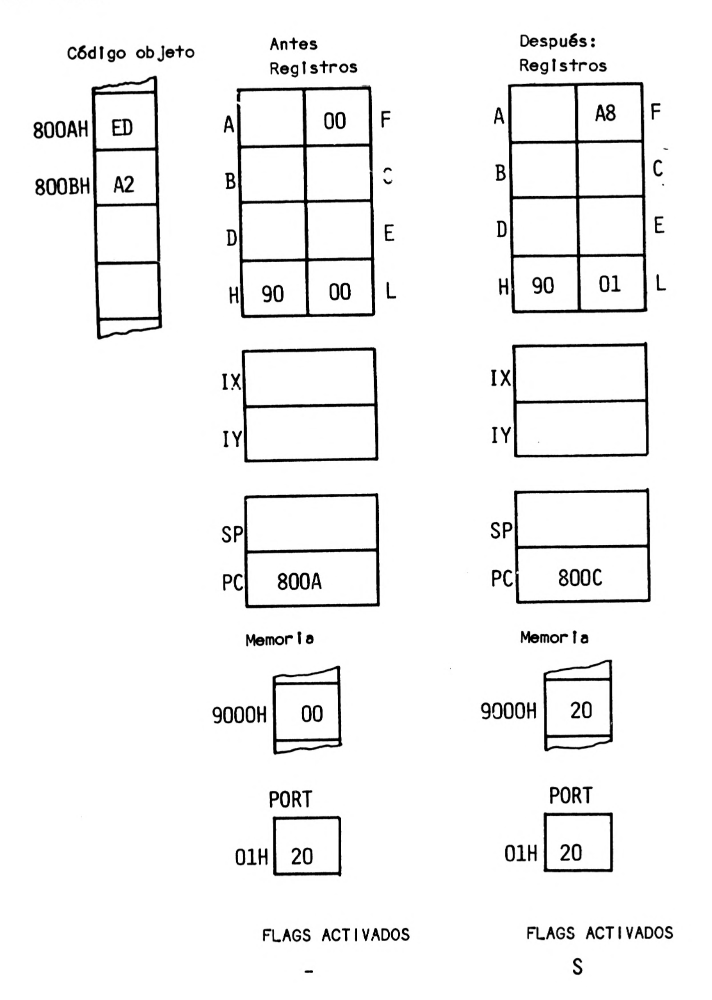
Tiempo de ejecución	: 4,00 µs
Frecuencia de reloj	: 4,00 MHz
Estados de reloj	: 16
Ciclos de máquina	: 4

Direccionamiento: Externo

Formato:

Byte 1)	1	1	1	0	1	1	0	1	EDH
Byte 2)	1	0	1	0	0	0	1	0	A2H

Ejemplo: INI


Código objeto
800AH
ED
800BH
A2
Antes
Registros
A
00
F
B
C
D
E
H
90
00
L
IX
IY
SP
PC
800A
Memoria
9000H
00
PORT
01H
20
FLAGS ACTIVADOS
-
Después:
Registros
A
A8
F
B
C
D
E
H
90
01
L
IX
IY
SP
PC
800C
Memoria
9000H
20
PORT
01H
20
FLAGS ACTIVADOS
S

INIR

Entrada de bloques con incremento.

Función:

(HL) --(C); B -- B - 1; HL -- HL + 1;
Repetición hasta B = 0.

Flags:

S	Z	-	H	-	P/V	N	C
?	1		?		?	1	

Descripción:

La posición de memoria (HL) direccionada indirectamente a través del par de registros HL será cargada mediante el operando (C) direccionado externamente a través del registro C. A continuación se decrementa B y se incrementa HL. El contenido del registro B se compara con cero. Unicamente cuando B sea diferente de cero, el contador del programa será decrementado en 2 y de este modo volverá a ejecutarse la instrucción.

Ejecución:

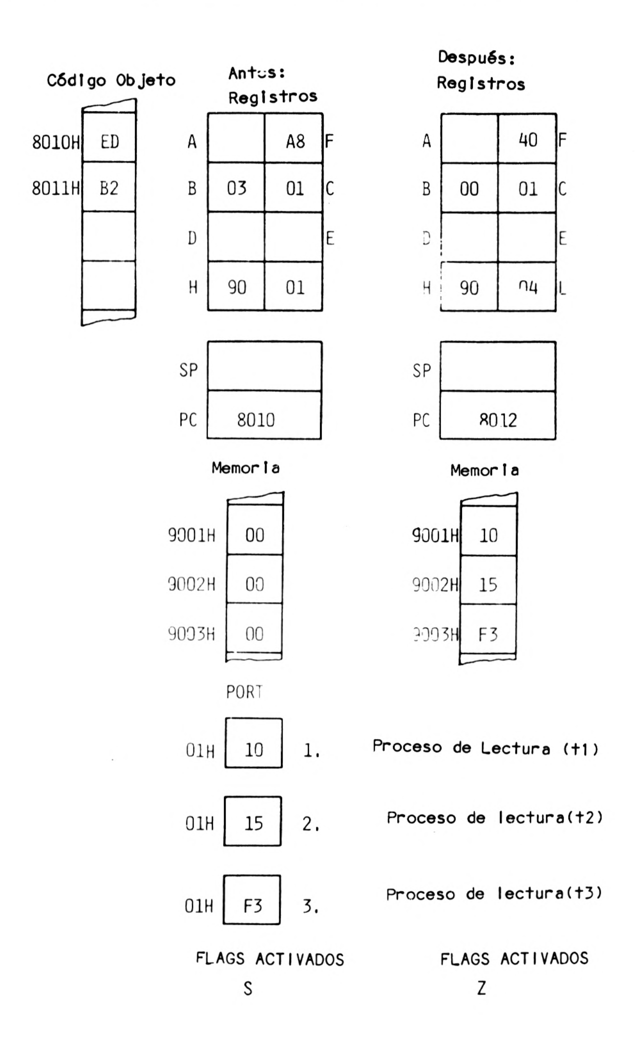
B = 0: 4 ciclos M; 16 estados T, 4 µs 4,00 MHz
B ≠ 0: 5 ciclos M; 21 estados T, 5,25 µs 4,00 MHz

Direccionamiento: Externo

Formato:

Byte 1)	1	1	1	0	1	1	0	1	EDH
Byte 2)	1	0	1	1	0	0	1	0	B2H

Ejemplo: INIR


Código Objeto
8010H
ED
8011H
B2
Antes:
Registros
A
A8
F
B
03
01
C
D
E
H
90
01
SP
PC
8010
Después:
Registros
A
40
F
B
00
01
C
D
E
H
90
04
L
SP
PC
8012
Memoria
9001H
00
9002H
00
9003H
00
Memoria
9001H
10
9002H
15
9003H
F3
PORT
01H
10
1.
Proceso de Lectura (+1)
01H
15
2.
Proceso de lectura(+2)
01H
F3
3.
Proceso de lectura(+3)
FLAGS ACTIVADOS
S
FLAGS ACTIVADOS
Z

JP (HL) **Bifurcación a la dirección (HL)**

Función: PC <——— HL

Flags: S Z - H - P/V N C
(ninguna influencia)

Descripción: El programa bifurca a la dirección contenida en el par de registros HL, el contenido del registro H se carga en la porción superior del PC y el contenido del registro L se carga en la porción inferior del contador del programa PC.

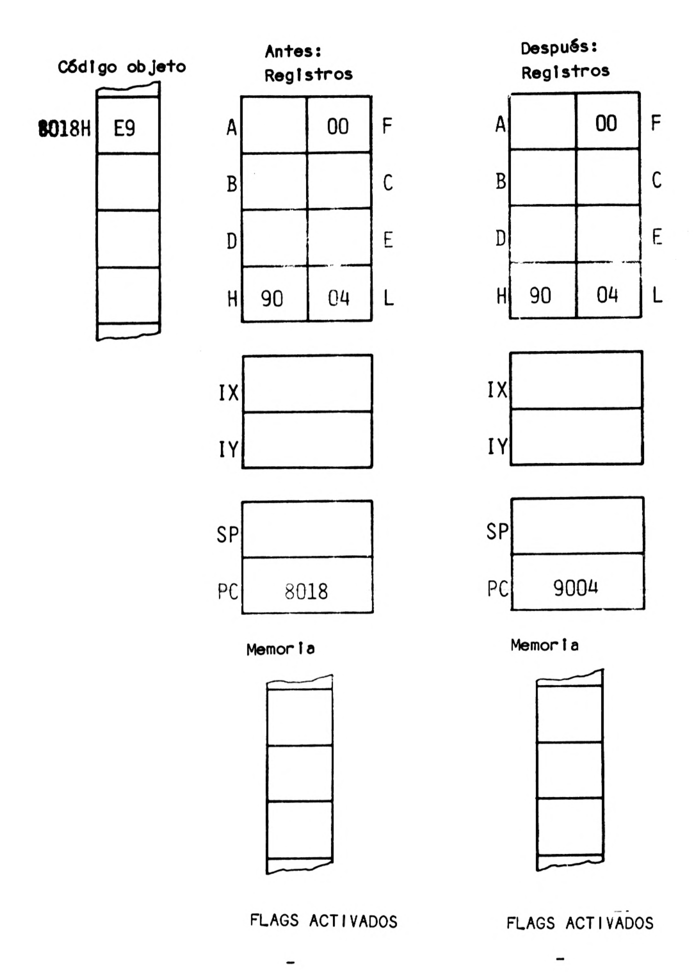
Ejecución:

Tiempo de ejecución	:	1,00 µs
Frecuencia de reloj	:	4,00 MHz
Estados de reloj	:	4
Ciclos de máquina	:	1

Direccionamiento: Implícito

Formato: 1 1 1 0 1 0 0 1 E9H

Ejemplo: JP (HL)



JP (IX) Bifurcación a la dirección (IX).

Función: PC <——— IX

Flags: S Z - H - P/V N C
(ninguna influencia)

Descripción: El programa bifurca a la dirección contenida en el registro de índices IX, el registro de índices se carga en el contador del programa.

Ejecución:

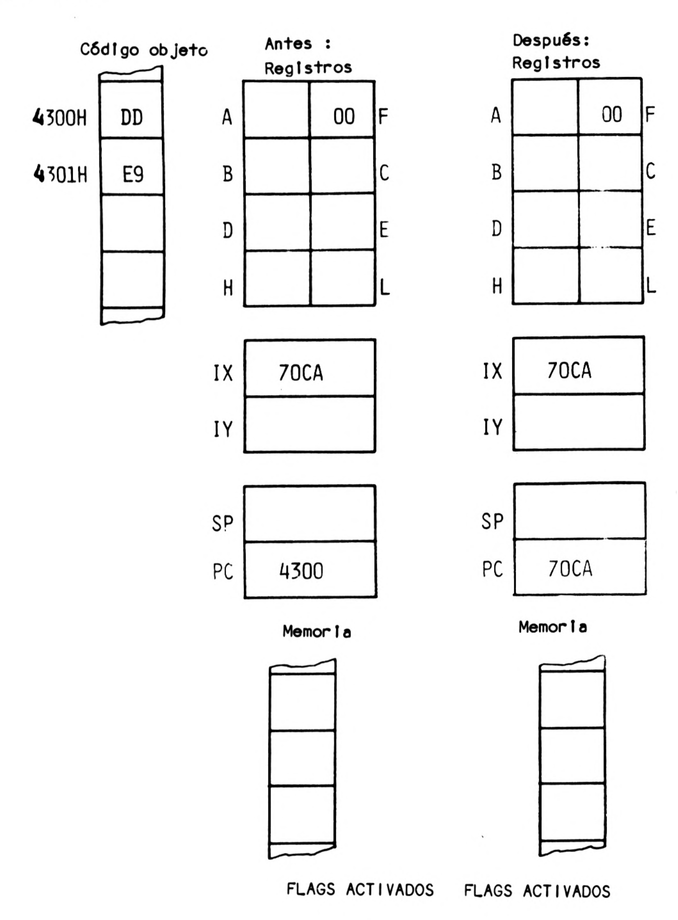
Tiempo de ejecución	:	2,00 µs
Frecuencia de reloj	:	4,00 MHz
Estados de reloj	:	8
Ciclos de máquina	:	2

Direccionamiento: Implícito

Formato:

Byte 1)	1	1	0	1	1	1	0	1	DDH
Byte 2)	1	1	1	0	1	0	0	1	E9H

Ejemplo: JP (IX)



JP (IY)

Bifurcación a la dirección (IY).

Función: PC <——— IY

Flags: S Z - H - P/V N C
(ninguna influencia)

Descripción: El programa bifurca a la dirección contenida en el registro de índices IY, en donde el registro de índices será cargado en el contador del programa.

Ejecución:

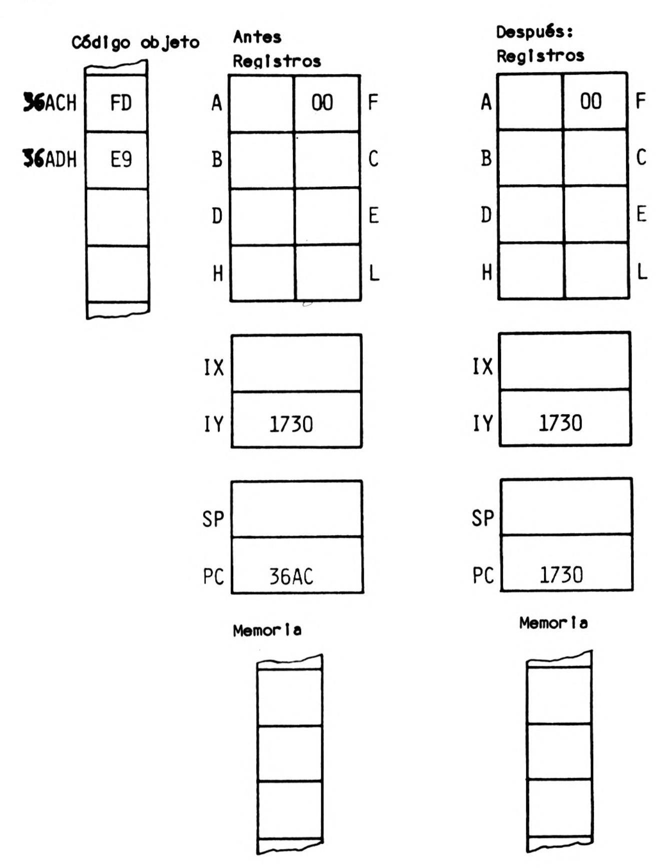
Tiempo de ejecución	:	2,00 µs
Frecuencia de reloj	:	4,00 MHz
Estados de reloj	:	8
Ciclos de máquina	:	2

Direccionamiento: Implícito

Formato:

Byte 1)	1	1	1	1	1	1	0	1	FDH
Byte 2)	1	1	1	0	1	0	0	1	E9H

Ejemplo: JP (IY)



FLAGS ACTIVADOS FLAGS ACTIVADOS
- -

JP cc,pq — Bifurcación condicional a la dirección pq.

Función: Cuando cc se cumple, entonces PC<—— pq.

Flags: S Z - H - P/V N C
(ninguna influencia)

Descripción: Cuando se cumple la condición cc, el programa bifurca a la dirección pq cargándose p en la mitad superior del PC y la q en la mitad inferior del PC. Si la condición no se cumple, el programa continuará con la próxima instrucción.

Ejecución:

Tiempo de ejecución	: 2,50 µs
Frecuencia de reloj	: 4,00 MHz
Estados de reloj	: 10
Ciclos de máquina	: 3

Direccionamiento: Inmediato

Formato:

Byte 1) 1 1-----cc----0 1 0

Byte 2) -----------p----------Porción inf. Direc.

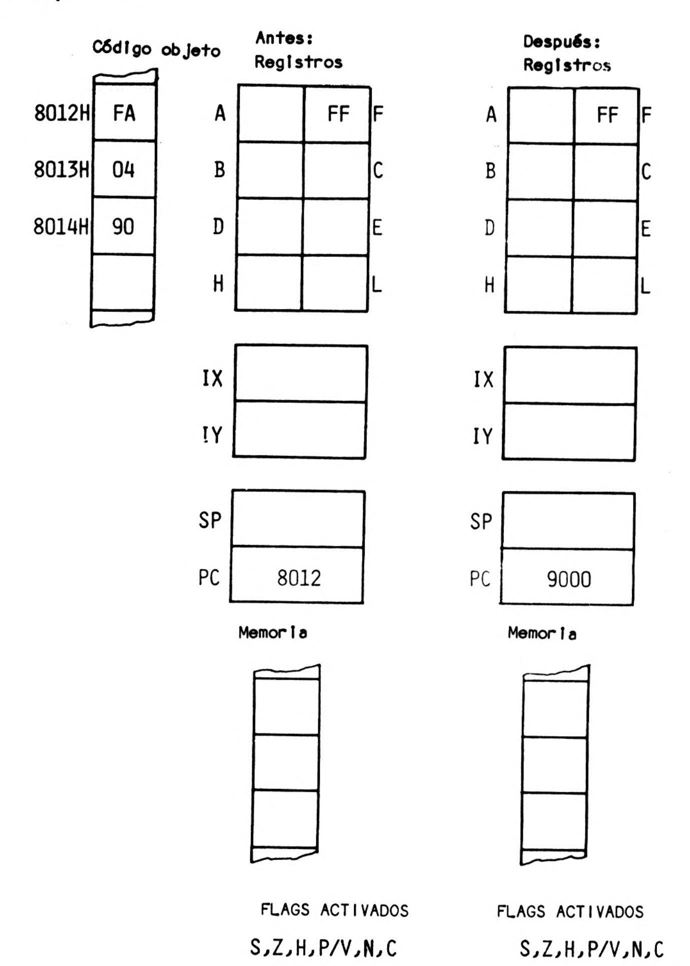
Byte 3) -----------q----------Porción super. Direc.

cc puede ser:

NZ	- 000	No cero
Z	- 001	Cero
NC	- 010	Ningún acarreo
C	- 011	Acarreo (Carry)
PO	- 100	Paridad impar
PE	- 101	Paridad par
P	- 110	Más
M	- 111	Menos

(Carry)

Ejemplo: JP M,9000H



JP pq — Bifurcación a la dirección pq.

Función: PC <——— pq

Flags: S Z - H - P/V N C
(ninguna influencia)

Descripción: El programa bifurca a la dirección pq donde p se carga en la mitad superior del PC y la q en la mitad inferior del PC.

Ejecución:

Tiempo de ejecución	:	2,50 µs
Frecuencia de reloj	:	4,00 MHz
Estados de reloj	:	10
Ciclos de máquina	:	3

Direccionamiento: Inmediato

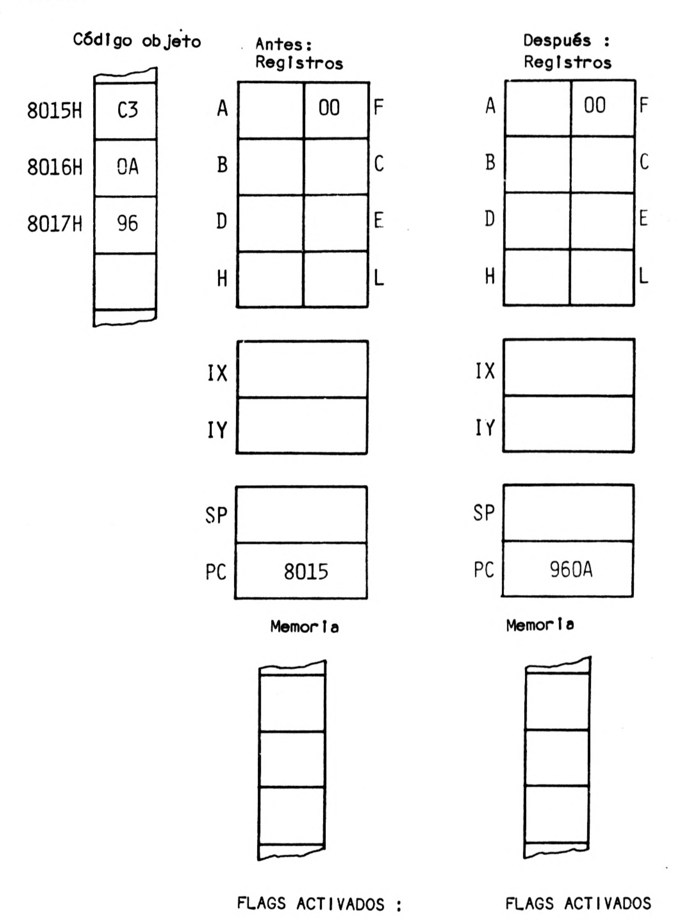
Formato:

Byte 1) 1 1 0 0 0 0 1 1 C3H

Byte 2) -----------q----------Porción infer. Direc.

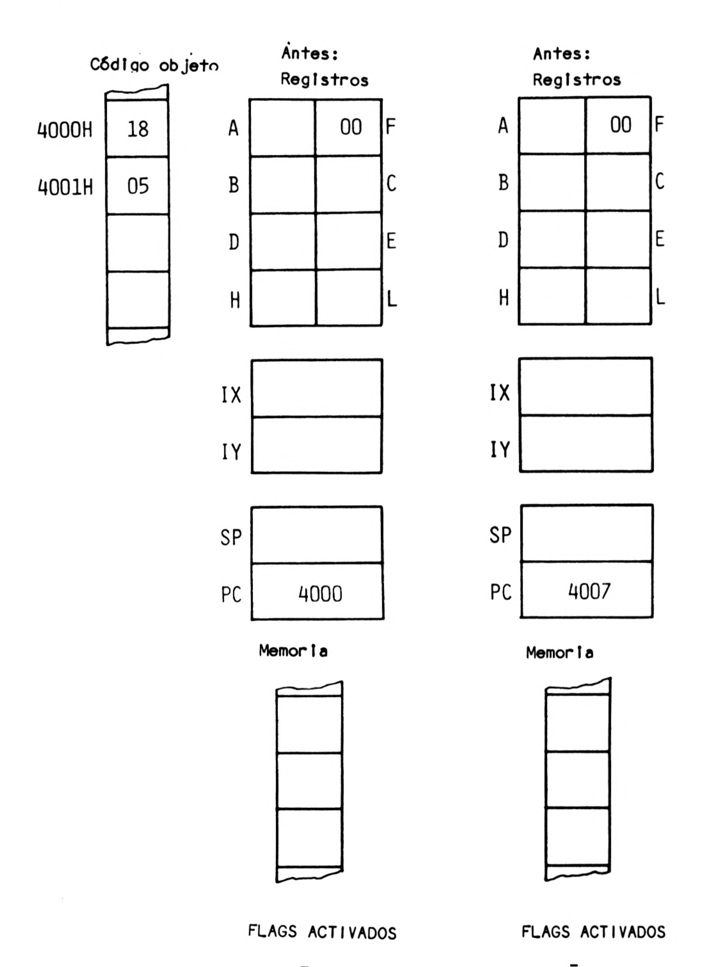
Byte 3) -----------p----------Porción super. Direc.

Ejemplo: JP 960AH



JR e	Bifurcación relativa a e.
Función:	PC <——— PC + e
Flags:	S Z - H - P/V N C (ninguna influencia)
Descripción:	El programa bifurca a la dirección resultante de la suma del contenido del contador del programa después de la ejecución de la instrucción (PC + 2) y del desplazamiento en forma de complemento binario. El campo efectivo de bifurcación se encontrará entre + 129 y - 126.
Ejecución:	Tiempo de ejecución : 3,00 µs Frecuencia de reloj : 4,00 MHz Estados de reloj : 12 Ciclos de máquina : 3
Direccionamiento:	Inmediato
Formato:	Byte 1) 0 0 0 1 1 0 0 0 18H Byte 2) ----------e-2---------Offset

Ejemplo: JR 05H



JR cc,e

Bifurcación relativa condicional a e.

Función: Si la cc se ha realizado, entonces
PC <—— PC + e

Flags: S Z - H - P/V N C
(ninguna influencia)

Descripción: Cuando se verifica la condición cc, el programa bifurca a la dirección resultante de la suma del contenido del contador del programa después de la ejecución de la instrucción (PC + 2) y del desplazamiento calculado por complemento binario. El campo efectivo de bifurcación se encontrará entre + 129 y - 126.

Ejecución:

	µs (4 MHz)	Estados T	Ciclos M
Cumple condición:	3	12	3
No cumple condición:	1,75	7	2

Direccionamiento: Inmediato

Formato:

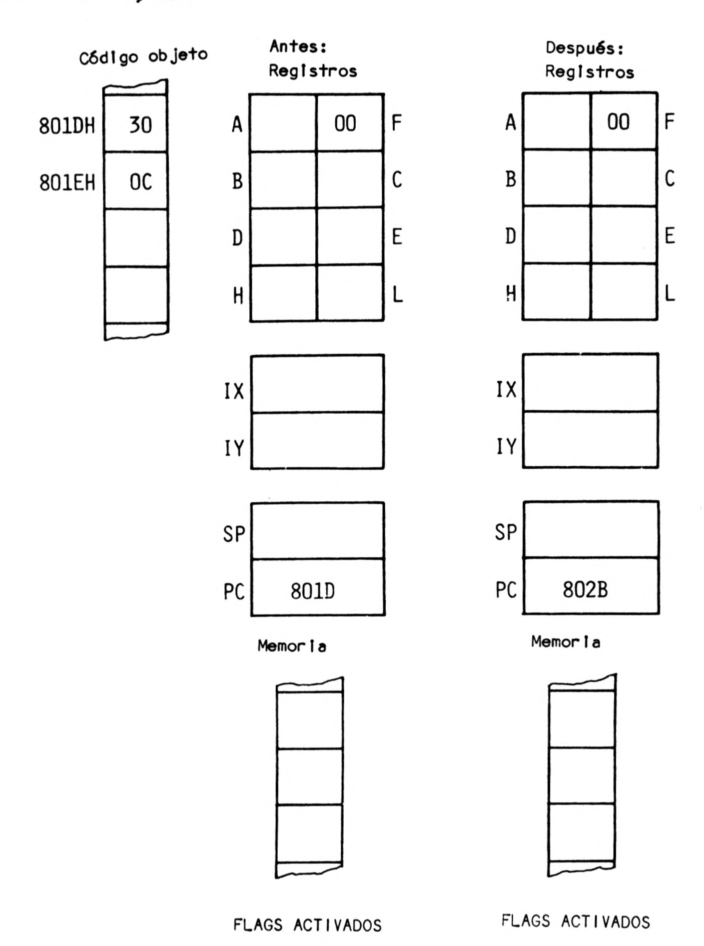
Byte 1) 0 0 1 c c 0 0 0

Byte 2) ----------e-2---------Offset

cc puede ser:

NZ - 00	No cero
Z - 01	Cero
NC - 10	Ningún acarreo (carry)
C - 11	Acarreo (carry)

Ejemplo: JR NC,OCH



LD A,(BC) — Carga el acumulador con el contenido de la posición de memoria direccionada indirectamente mediante el par de registros BC.

Función: A <——— (BC)

Flags: S Z - H - P/V N C
(ninguna influencia)

Descripción: El acumulador A se cargará con el operando (BC) direccionado indirectamente a través del par de registros BC.

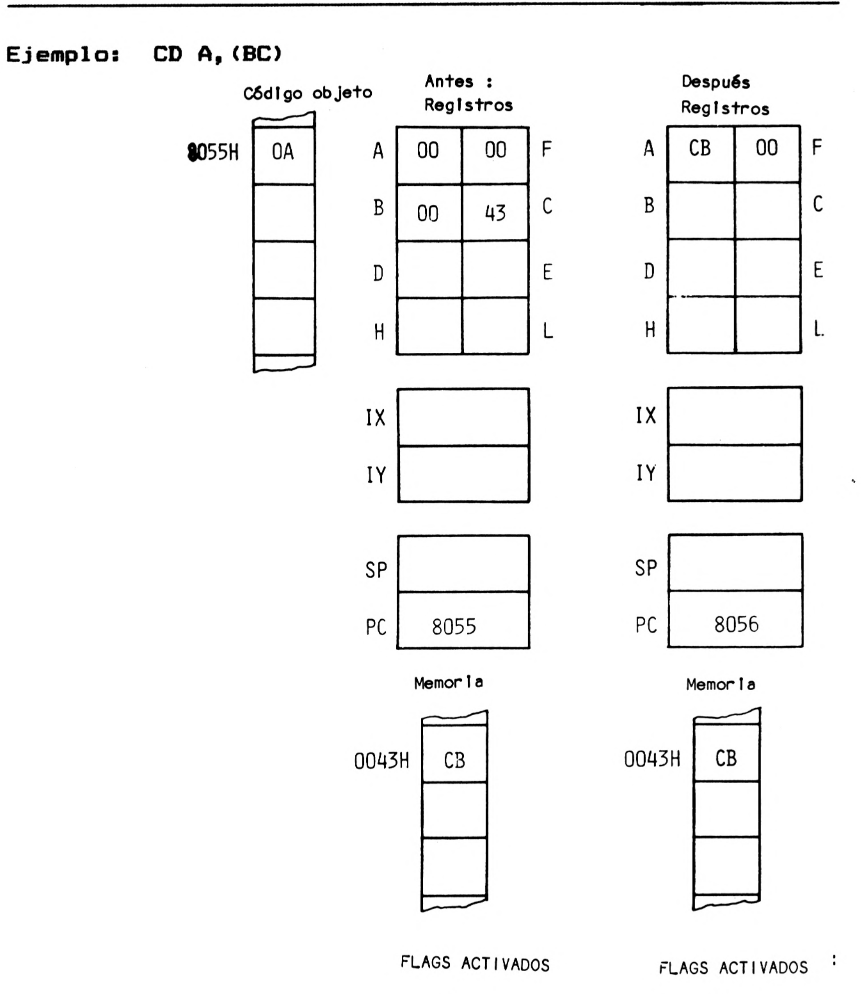
Ejecución:

Tiempo de ejecución	:	1,75 µs
Frecuencia de reloj	:	4,00 MHz
Estados de reloj	:	7
Ciclos de máquina	:	2

Direccionamiento: Indirecto

Formato: 0 0 0 0 1 0 1 0 0AH

Ejemplo: CD A,(BC)



LD A,(DE) Carga el acumulador con el contenido de la posición de memoria direccionada indirectamente mediante el par de registros DE.

Función: A <——— (DE)

Flags: S Z - H - P/V N C
(ninguna influencia)

Descripción: El acumulador A se cargará con el operando (DE) direccionado indirectamente a través del par de registros DE.

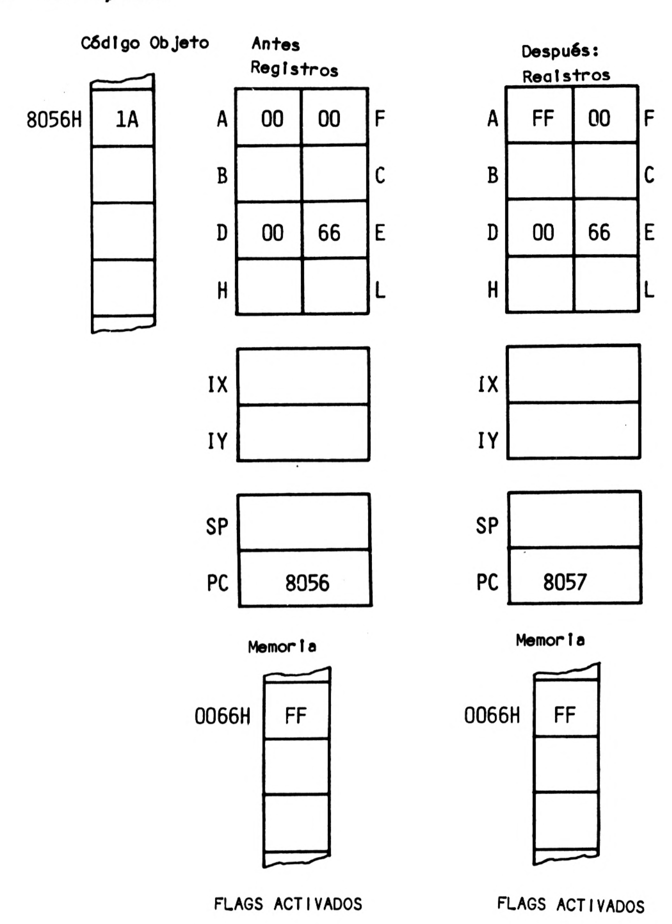
Ejecución:

Tiempo de ejecución	:	1,75 µs
Frecuencia de reloj	:	4,00 MHz
Estados de reloj	:	7
Ciclos de máquina	:	2

Direccionamiento: Indirecto

Formato: 0 0 0 1 1 0 1 0 1AH

Ejemplo: LD A,(DE)



LD A,I — Carga el acumulador con el contenido del registro de vectores del nterrupt I.

Función: A <——— 1

Flags:

S	Z	-	H	-	P/V	N	C
X	X		0		X	0	

Activ. por el contenido de IFF2.

Descripción: El acumulador A se cargará con el contenido del registro I. El registro I de interrupt contiene la mitad superior del vector de interrupt que se genera al ocurrir una interrupción en modalidad 2 de interrupt (IM 2). (La mitad inferior se envía al bus de datos mediante el componente que provoca la interrupción.)

Ejecución:

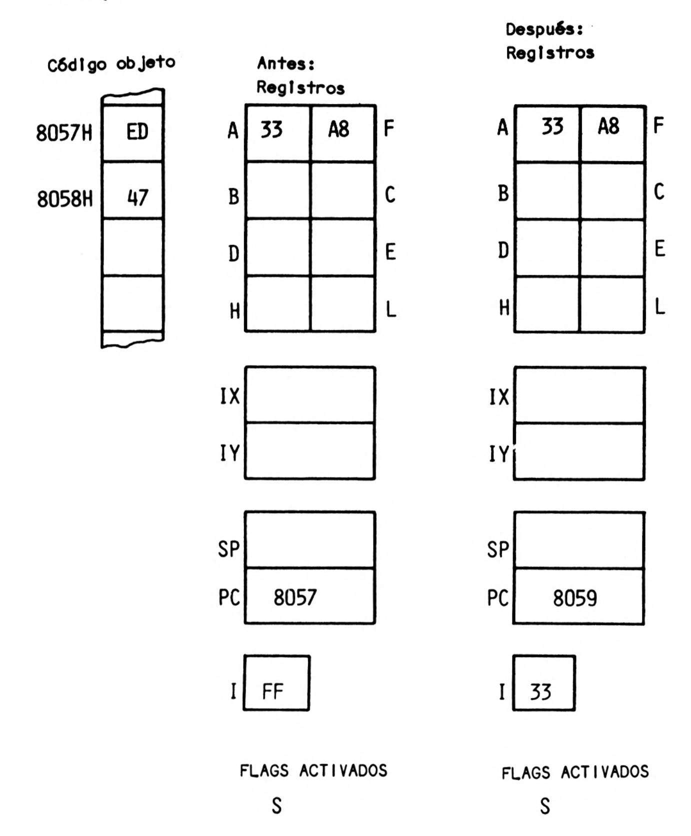
Tiempo de ejecución : 2,25 µs
Frecuencia de reloj : 4,00 MHz
Estados de reloj : 9
Ciclos de máquina : 2

Direccionamiento: Implícito

Formato:

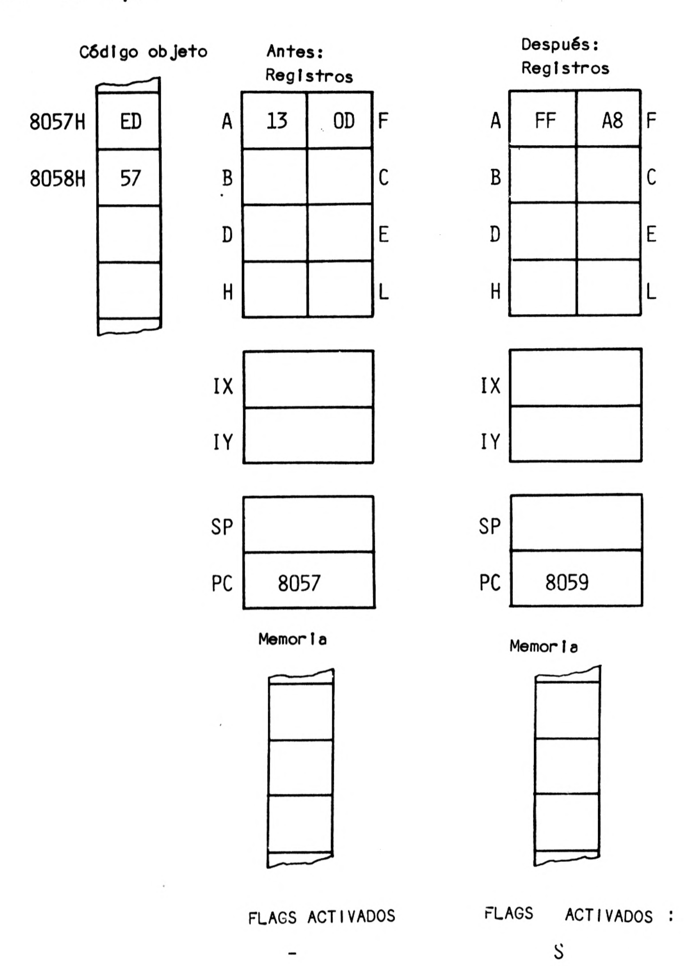
Byte 1)	1	1	1	0	1	1	0	1	EDH
Byte 2)	0	1	0	1	0	1	1	1	57H

Ejemplo: LD I,A



LD A,(nn)	**Carga el acumulador con el contenido de la posición de memoria (nn).**
Función:	A <——— (nn)
Flags:	S Z - H - P/V N C (ninguna influencia)
Descripción:	El acumulador A se cargará con el operando (nn) direccionado directamente a través de la dirección nn de 16 Bits.
Ejecución:	Tiempo de ejecución : 3,25 µs Frecuencia de reloj : 4,00 MHz Estados de reloj : 13 Ciclos de máquina : 4
Direccionamiento:	Directo
Formato:	Byte 1) 0 0 1 1 1 0 1 0 3AH Byte 2) -----------n----------Porción inf. Direc. Byte 3) -----------n----------Porción super. Direc.

Ejemplo: LD A,I



FLAGS ACTIVADOS

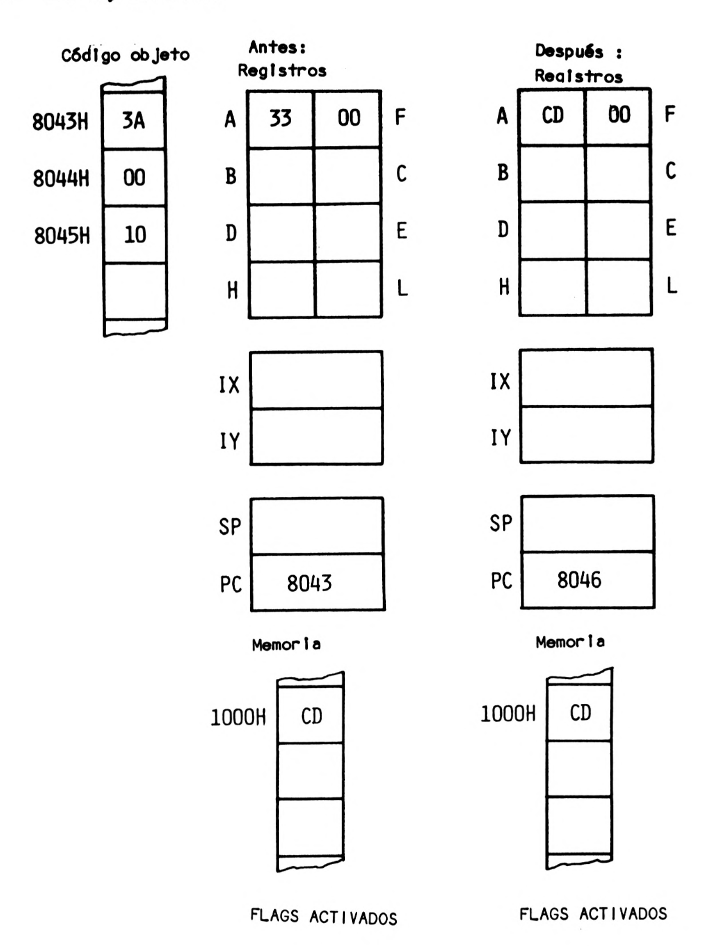
-

FLAGS ACTIVADOS :

S

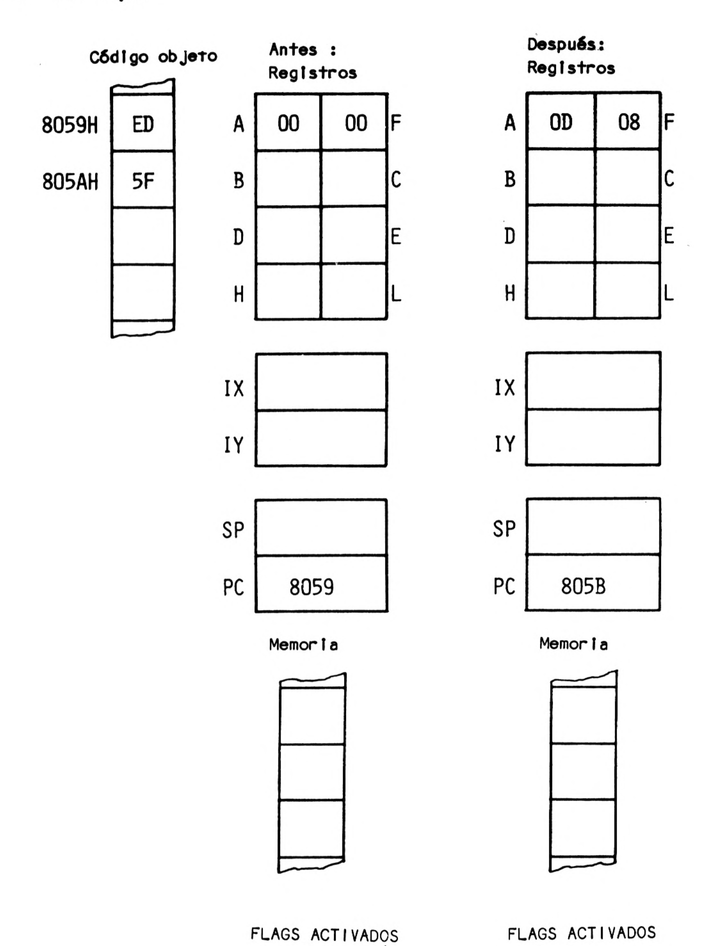
LD A,R	Carga el acumulador con el contenido del registro de memoria de refresco R.
Función:	A -- R
Flags:	S Z - H - P/V N C
	X X 0 X 0
	Activado por el contenido del IFF2.
Descripción:	El acumulador A se cargará con el contenido del registro R. El registro de refresco R contiene la dirección de refresco para la memoria dinámica eventualmente conectada.
Ejecución:	Tiempo de ejecución : 2,25 μs
	Frecuencia de reloj : 4,00 MHz
	Estados de reloj : 9
	Ciclos de máquina : 2
Direccionamiento:	Implícito
Formato:	Byte 1) 1 1 1 0 1 1 0 1 EDH
	Byte 2) 0 1 0 1 1 1 1 1 5FH

Ejemplo: LD A,(1000H)



LD(BC),A	Carga la célula de memoria (BC) direccionada indirectamente con el contenido del acumulador.
Función:	(BC) <——— A
Flags:	S Z - H - P/V N C (ninguna influencia)
Descripción:	La posición de memoria (BC) direccionado indirectamente a través del par de registros BC será cargada con el contenido del acumulador A.
Ejecución:	Tiempo de ejecución : 1,75 µs Frecuencia de reloj : 4,00 MHz Estados de reloj : 7 Ciclos de máquina : 2
Direccionamiento:	Indirecto
Formato:	0 0 0 0 0 0 1 0 02H

Ejemplo: LD A,R



<u>LD (DE),A</u> Carga la célula de memoria (DE) direccionada indirectamente con el contenido del acumulador.

Función: (DE) <——— A

Flags: S Z - H - P/V N C
(ninguna influencia)

Descripción: La célula de memoria (DE) direccionada indirectamente a través del par de registros DE será cargada con el contenido del acumulador A.

Ejecución:

Tiempo de ejecución	:	1,75 µs
Frecuencia de reloj	:	4,00 MHz
Estados de reloj	:	7
Ciclos de máquina	:	2

Direccionamiento: Indirecto

Formato: 0 0 0 1 0 0 1 0 12H

Ejemplo: LD (BC),A

	Antes:	Después:

Código objeto

02

Antes: Registros

A	33	00	F
B	F4	00	C
D			E
H			L
IX			
IY			
SP			
PC	802A		

Después: Registros

A	33	00	F
B	F4	00	C
D			E
H			L
IX			
IY			
SP			
PC	802B		

Memoria (Antes)

F400H	00

Memoria (Después)

F400H	33

Flags activados: (Antes)

Flags activados: (Después)

<u>LD (HL),n</u> — Carga los datos n inmediatos en la célula de memoria (HL) direccionados indirectamente.

Función: (HL) <——— n

Flags: S Z - H - P/V N C
(ninguna influencia)

Descripción: La posición de memoria (HL) direccionado indirectamente a través del par de registros HL será cargada con los datos n que se encuentran inmediatamente después de la parte operacional de la instrucción.

Ejecución:

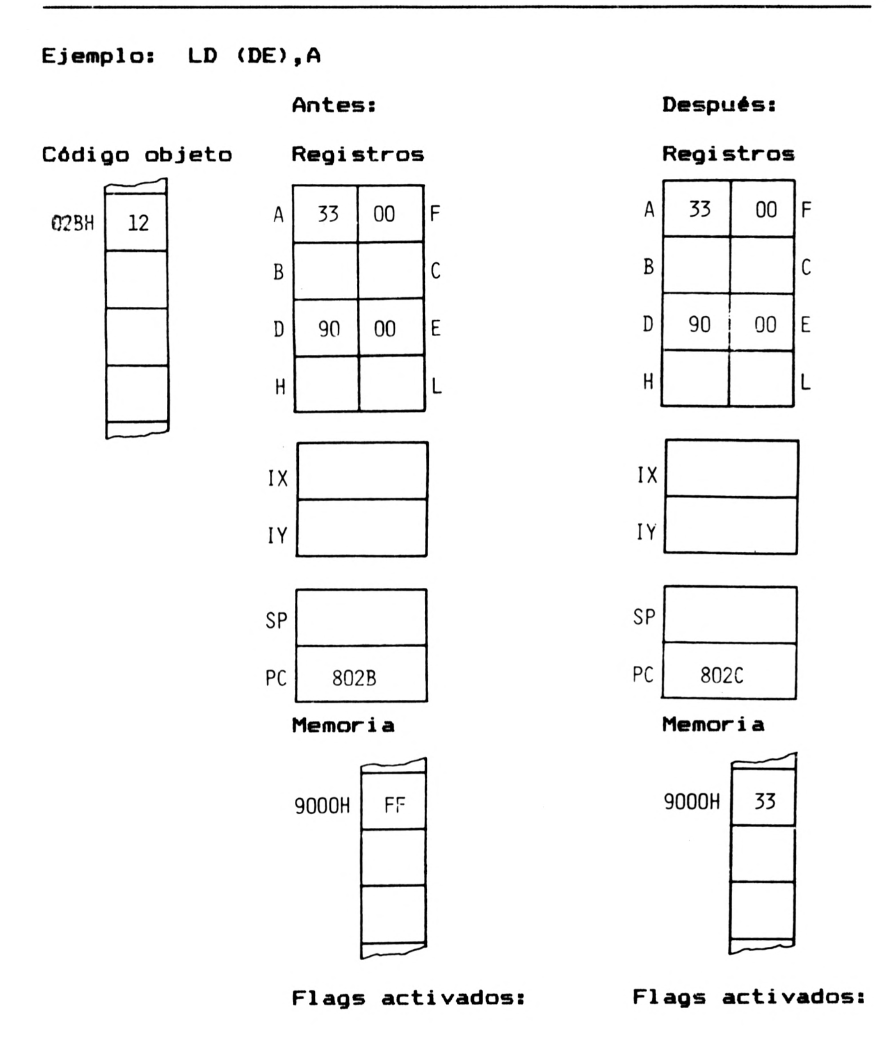
Tiempo de ejecución	: 2,50 µs
Frecuencia de reloj	: 4,00 MHz
Estados de reloj	: 10
Ciclos de máquina	: 3

Direccionamiento: Inmediato/indirecto

Formato:

Byte 1)	0	0	1	1	0	1	1	0	36H
Byte 2)	-----------n----------								Datos inmed.

Ejemplo: LD (DE),A



<u>LD dd,nn</u> — **Carga el par de registros dd con los datos nn inmediatos**

Función: dd <——— nn

Flags: S Z - H - P/V N C
(ninguna influencia)

Descripción: El par de registros dd se cargará con los datos inmediatos nn de 16 bits. Los 8 primeros bits siguientes a la parte operacional de la instrucción se cargan en la porción inferior del par de registros dd y los 8 bits restantes se cargan en la porción superior del par de registros asignados.

Ejecución:

Tiempo de ejecución	:	2,50 µs
Frecuencia de reloj	:	4,00 MHz
Estados de reloj	:	10
Ciclos de máquina	:	3

Direccionamiento: Inmediato

Formato:

```
Byte 1) 0  0  d  d  0  0  0  1

Byte 2) ----------n-----------Mitad
                        inferior de
                        Datos inmed.
Byte 3) ----------n-----------Mitad
                        superior
                        Datos inmed.
```

dd puede ser:

BC - 00	HL - 10
DE - 01	SP - 11

Ejemplo: LD (HL),3H

Antes:

Código objeto

802CH	36
802CH	03

Registros

A		00	F
B			C
D			E
H	98	76	L

IX	
IY	

SP	
PC	802C

Memoria

9876H	00

Flags activados:

Después:

Registros

A		00	F
B			C
D			E
H	98	76	L

IX	
IY	

SP	
PC	802E

Memoria

9876H	03

Flags activados:

LD dd,(nn)

Carga el par de registros dd con el contenido de la célula de memoria nn direccionada directamente.

Función: $dd_{inferior} \leftarrow (nn)$; $dd_{superior} \leftarrow (nn+1)$

Flags: S Z - H - P/V N C
(ninguna influencia)

Descripción: El par de registros dd se carga con el operando de 16 bits direccionados directamente. dado que el par de registros contiene 2 bytes y cada posición de memoria corresponde a 1 byte, se carga en primer lugar la dirección de memoria direccionada por nn en el registro inferior del par de registros utilizados. A continuación se incrementa el contador de programa y se carga la posición de memoria correspondiente a (nn + 1) en el registro superior del par de registros dd.

Ejecución:

Tiempo de ejecución	:	5,00 µs
Frecuencia de reloj	:	4,00 MHz
Estados de reloj	:	20
Ciclos de máquina	:	6

Direccionamiento: Directo

Formato:										
	Byte 1)	1	1	1	0	1	1	0	1	EDH
	Byte 2)	0	1	d	d	1	0	1	1	
	Byte 3)	----------n----------								Porción infer. direc.
	Byte 4)	----------n----------								Porción super. direc.

dd puede ser:

BC - 00	HL - 10
DE - 01	SP - 11

Ejemplo: LD BC,F4E2H

Código objeto

Dirección	Contenido
8024H	01
8025H	E2
8026H	F4

Antes:

Registros

A		0D	F
B	00	01	C
D			E
H			L

IX	
IY	

SP	
PC	8024

Memoria

Flags activados:

Después:

Registros

A		00	F
B	F4	E2	C
D			E
H			L

IX	
IY	

SP	
PC	8027

Memoria

Flags activados:

LD HL,(nn) — Carga el par de registros HL desde la célula de memoria nn.

Función: L <—— (nn); H <—— (nn + 1)

Flags: S Z - H - P/V N C
(ninguna influencia)

Descripción: El par de registros HL se cargará mediante el operando de 16 Bits direccionado directamente a través de la dirección nn. Dado que el par de registros contiene 2 bytes y cada posición de memoria corresponde a 1 byte, se carga en primer lugar la posición de memoria direccionada por nn en el registro L. A continuación se incrementa el contador de programa y la posición de memoria direccionada por (nn + 1) se carga en el registro H.

Ejecución:

Tiempo de ejecución	:	4,00 µs
Frecuencia de reloj	:	4,00 MHz
Estados de reloj	:	16
Ciclos de máquina	:	5

Direccionamiento: Directo

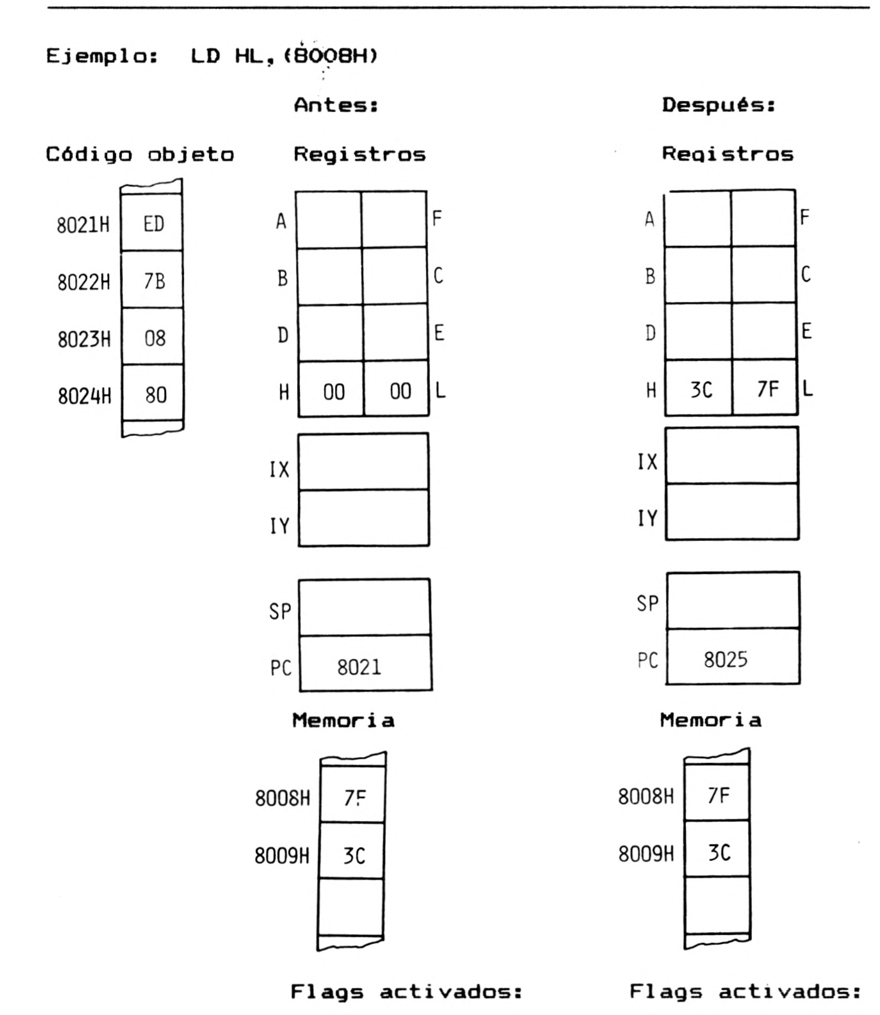
Formato:

Byte 1) 0 0 1 0 1 0 1 0 2AH

Byte 2) -----------n----------Porción inf. Direc.

Byte 3) -----------n----------Porción super. Direc.

Ejemplo: LD HL,(8008H)



Flags activados: Flags activados:

LD (HL),r

Carga la célula de memoria (HL) direccionada indirectamente con el contenidodel registro r.

Función: (HL) <——— r

Flags: S Z - H - P/V N C
(ninguna influencia)

Descripción: La posición de memoria (HL) direccionada indirectamente a través del par de registros HL se carga con el contenido del registro r.

Ejecución:

Tiempo de ejecución : 1,75 µs
Frecuencia de reloj : 4,00 MHz
Estados de reloj : 7
Ciclos de máquina : 2

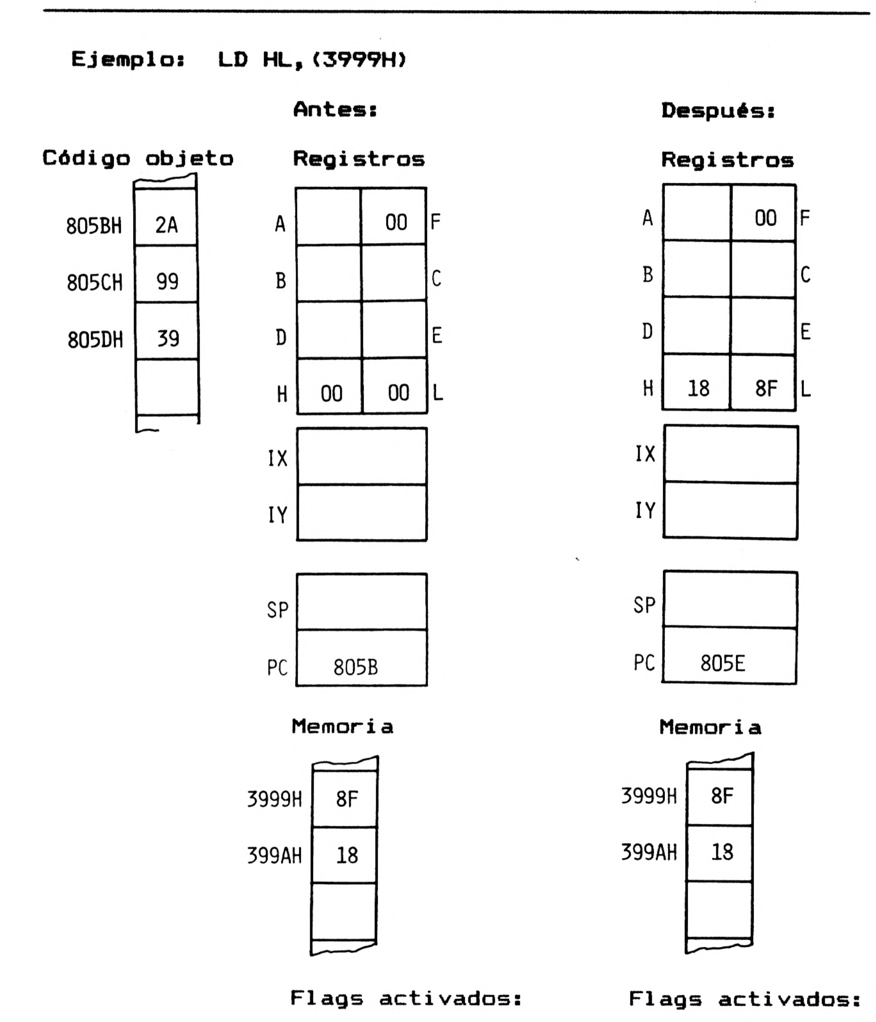
Direccionamiento: Indirecto

Formato: 0 1 1 1 0-----r-----

r puede ser:

A - 111	E - 011
B - 000	H - 100
C - 001	L - 101
D - 010	

Ejemplo: LD HL,(3999H)



Flags activados: Flags activados:

LD I,A — **Carga el registro I del interrupt con el contenido del acumulador.**

Función: I <——— A

Flags: S Z - H - P/V N C
(ninguna influencia)

Descripción: El registro I será cargado con el contenido del acumulador A. El registro de vector de interrupt I contiene la mitad superior del vector de interrupt que se creará en la modalidad 2 del Interrupt (IM 2) en el caso de presentarse una interrupción. (La mitad inferior envía el componente a ser interrum pido al bus de datos.)

Ejecución:

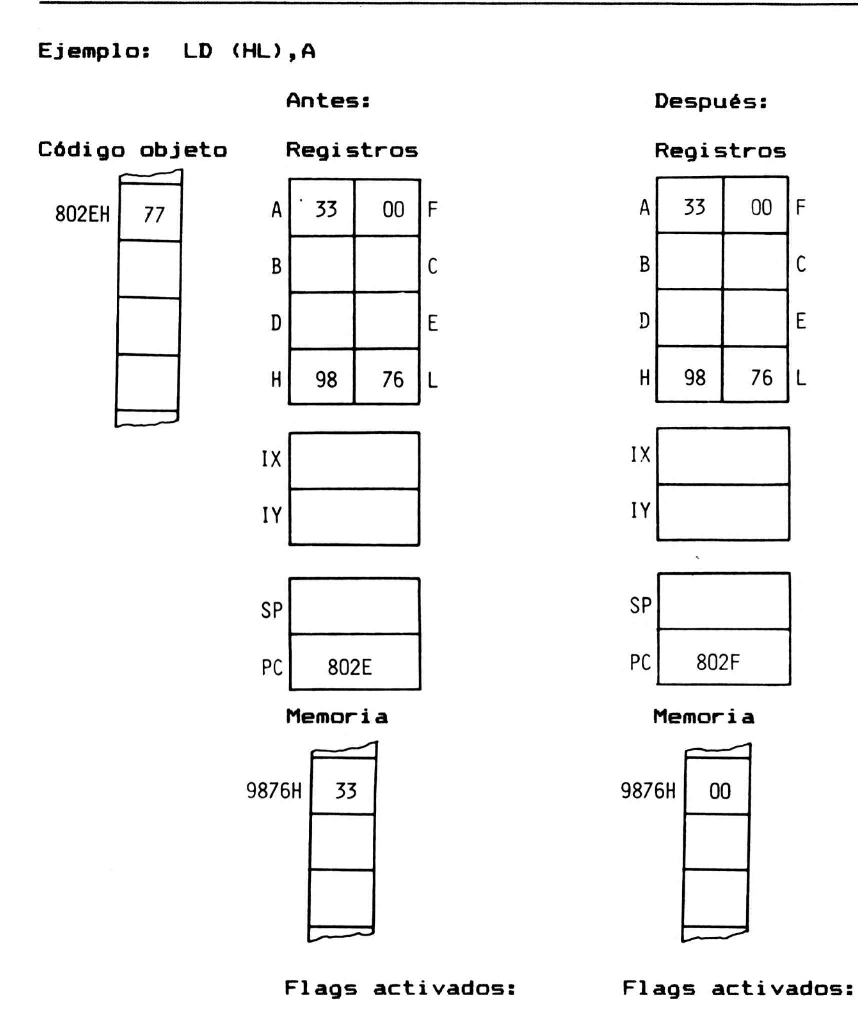
Tiempo de ejecución	:	2,25 µs
Frecuencia de reloj	:	4,00 MHz
Estados de reloj	:	9
Ciclos de máquina	:	2

Direccionamiento: Implícito

Formato:

Byte 1)	1	1	1	0	1	1	0	1	EDH
Byte 2)	0	1	0	0	0	1	1	1	47H

Ejemplo: LD (HL),A



Flags activados: Flags activados:

<u>LD IX,nn</u> Carga el par de registros IX con los datos nn inmediatos.

Función: IX <—— nn

Flags: S Z - H - P/V N C
(ninguna influencia)

Descripción: El par de registros IX se carga con los datos inmediatos de 16 bits nn. Los 8 primeros bits del dato inmediato se cargan en la porción inferior del par de registros y los 8 bits restantes en la porción superior del par de registros.

Ejecución:

Tiempo de ejecución	:	3,50 µs
Frecuencia de reloj	:	4,00 MHz
Estados de reloj	:	14
Ciclos de máquina	:	4

Direccionamiento: Inmediato

Formato:

Byte 1)	1	1	0	1	1	1	0	1	DDH
Byte 2)	0	0	1	0	0	0	0	1	21H
Byte 3)	-----------n----------								(*)
Byte 4)	-----------n----------								(*)

(*) Datos inmediatos mitad inferior.
Datos inmediatos mitad superior.

Ejemplo: LD IX,FF43H

Antes:

Código objeto

805EH	DD
805FH	21
8060H	43
8061H	FF

Registros

A		08	F
B			C
D			E
H			L

IX	0000
IY	

SP	
PC	805E

Memoria

Flags activados:

Después:

Registros

A		08	F
B			C
D			E
H			L

IX	FF43
IY	

SP	
PC	8062

Memoria

Flags activados:

LD IX,(nn) — **Carga el par de registros IX de la célula de memoria nn.**

Función: $IX_{inferior}$ <—— (nn); $IX_{superior}$ <—— (nn+1)

Flags: S Z - H - P/V N C
(ninguna influencia)

Descripción: El registro de índices IX será cargado con el operando de 16 bits direccionado directamente a través de la dirección nn. Dado que el par de registros corresponde a 2 bytes y cada posición de memoria a 1 byte se carga en primer lugar la posición de memoria direccionada mediante nn en el registro inferior. A continuación se incrementa el contador de programa, y se carga en el registro superior la posición de memoria direccionada por (nn + 1).

Ejecución:

Tiempo de ejecución	:	5,00 µs
Frecuencia de reloj	:	4,00 MHz
Estados de reloj	:	20
Ciclos de máquina	:	6

Direccionamiento: Directo

Formato:

Byte 1) 1 1 0 1 1 1 0 1 DDH

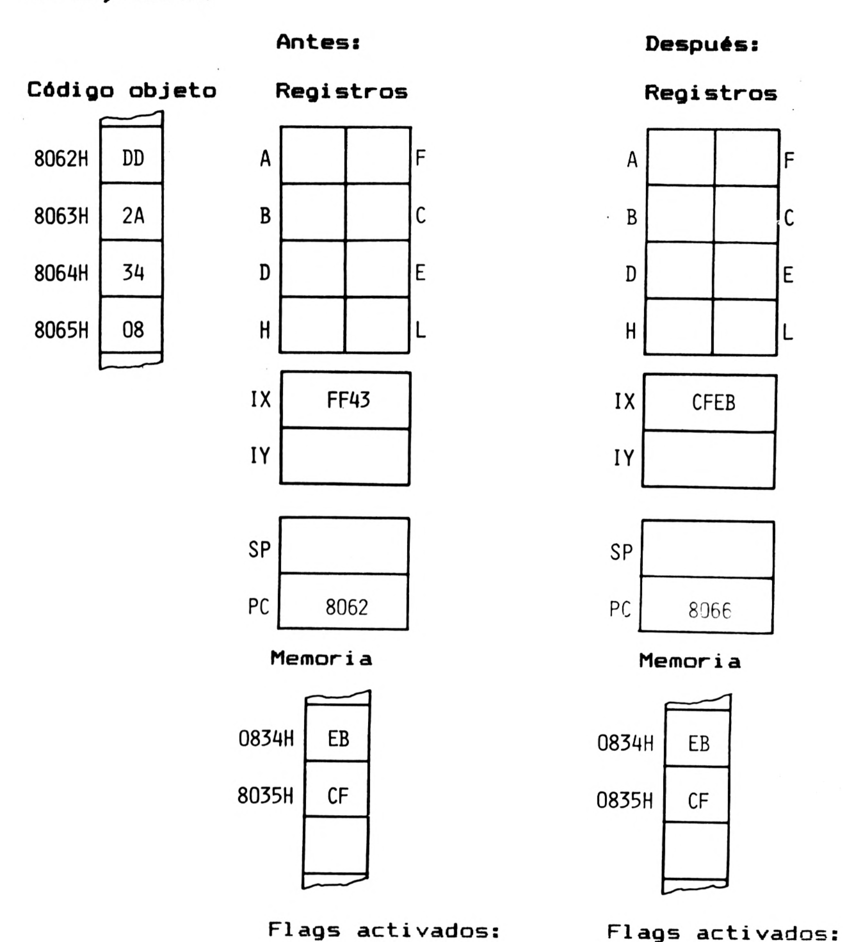
Byte 2) 0 0 1 0 1 0 1 0 2AH

Byte 3) ----------n----------- (*)

Byte 4) ----------n----------- (*)

(*) Dirección, dirección mitad infer., mitad superior.

LD IX,(0834H)



Flags activados: Flags activados:

<u>LD (IX + d),n</u>

Carga la célula de memoria (IX + d) direccionada indexadamente con los datos n inmediatos.

Función: (IX + d) <—— n

Flags: S Z - H - P/V N C
(ninguna influencia)

Descripción: La posición de memoria (IX + d) direccionada por índice a través del par de registros IX y el desplazamiento d, se carga con el dato inmediato de 8 bits n.

Ejecución:

Tiempo de ejecución	:	4,75 µs
Frecuencia de reloj	:	4,00 MHz
Estados de reloj	:	19
Ciclos de máquina	:	5

Direccionamiento: Indexado/inmediato

Formato:

Byte 1) 1 1 0 1 1 1 0 1 DDH

Byte 2) 0 0 1 1 0 1 1 0 36H

Byte 3) ----------d-----------Offset

Byte 4) ----------n-----------Datos inmed.

Ejemplo: LD (IX + 2),2

Antes:

Código objeto

Dirección	Contenido
8035H	DD
8036H	36
8037H	02
8038H	02

Registros

A		00	F
B			C
D			E
H			L

IX	A000
IY	

SP	
PC	8035

Memoria

A002H	FF

Flags activados:

Después:

Registros

A		00	F
B			C
D			E
H			L

IX	A000
IY	

SP	
PC	8039

Memoria

A002H	02

Flags activados:

LD (IX + d),r — **Carga la célula de memoria (IX + d) direccionada indexadamente con el contenido del registro r.**

Función: (IX + d) <—— r

Flags: S Z - H - P/V N C
(ninguna influencia)

Descripción: La posición de memoria direccionada por índices mediante el par de registros IX y el desplazamiento d (IX + d), se carga con el contenido del registro r.

Ejecución:

Tiempo de ejecución	: 4,75 µs
Frecuencia de reloj	: 4,00 MHz
Estados de reloj	: 19
Ciclos de máquina	: 5

Direccionamiento: Indexado

Formato:

Byte 1) 1 1 1 1 1 1 0 1 DDH

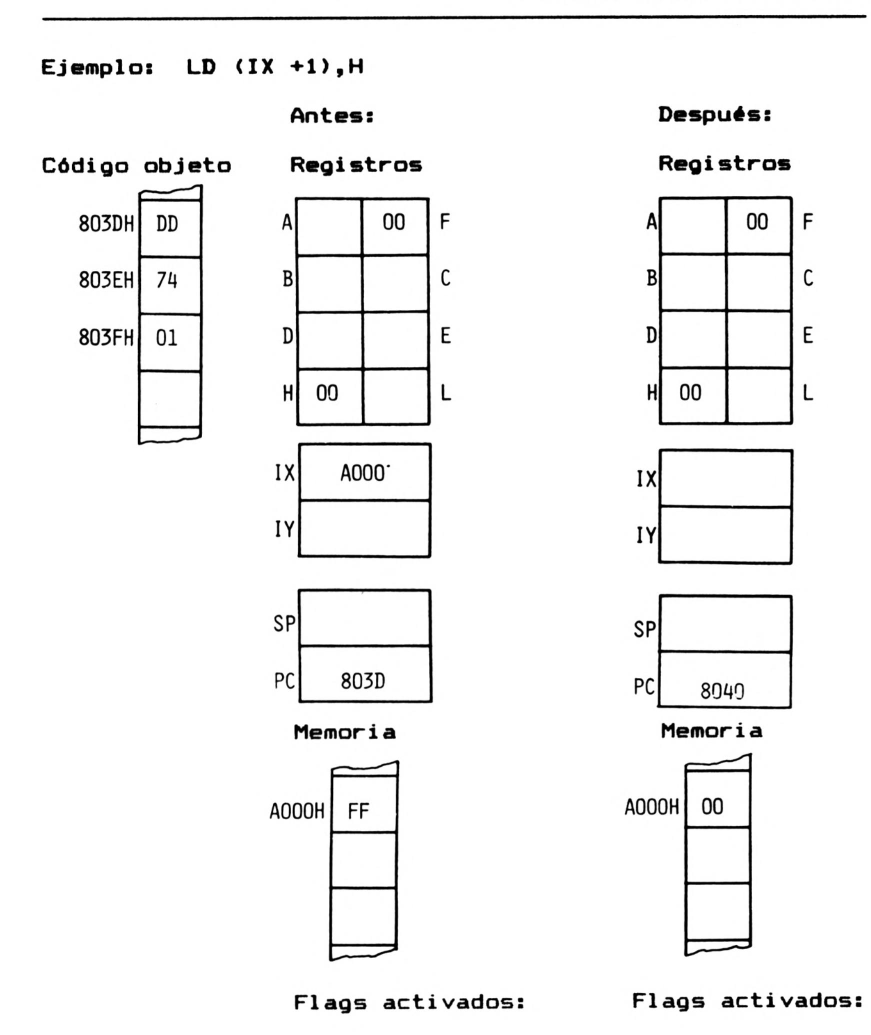
Byte 2) 0 1 1 1 0-----r---Offset

Byte 3) -----------d----------Offset

r puede ser:

A - 111	E - 011
B - 000	H - 100
C - 001	L - 101
D - 010	

Ejemplo: LD (IX +1),H



Flags activados: Flags activados:

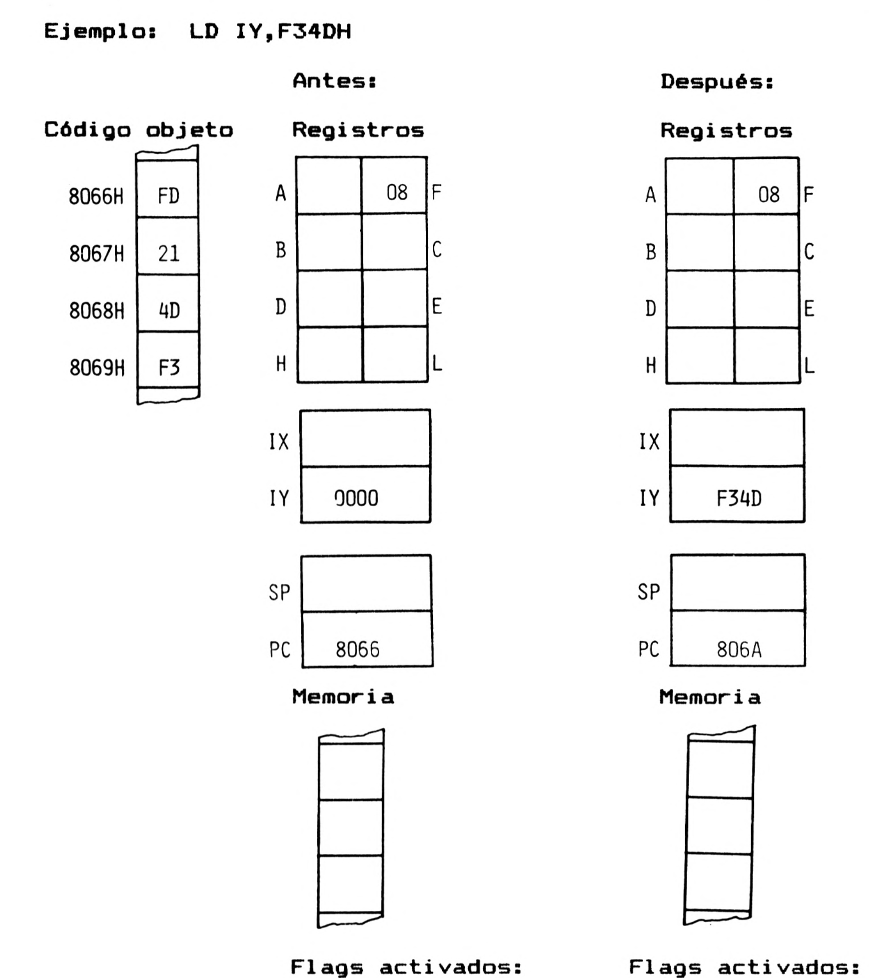
LD IY,nn	**Carga el par de registros IY con los datos nn inmediatos.**
Función:	IY <—— nn
Flags:	S Z - H - P/V N C (ninguna influencia)
Descripción:	El par de registros de índices IY se cargan con el dato inmediato nn de 16 bits. Los datos inmediatos se encuentran a continuación de la parte operacional de la instrucción. Los 8 primeros bits del dato inmediato se cargan en la porción inferior de los registros IY y los 8 bits restantes en la porción superior del registro IY.
Ejecución:	Tiempo de ejecución : 3,50 µs Frecuencia de reloj : 4,00 MHz Estados de reloj : 14 Ciclos de máquina : 4
Direccionamiento:	Inmediato

Formato:

Byte 1)	1	1	1	1	1	1	0	1	FDH
Byte 2)	0	0	1	0	0	0	0	1	21H
Byte 3)	-----------n----------								(*)
Byte 4)	-----------n----------								(*)

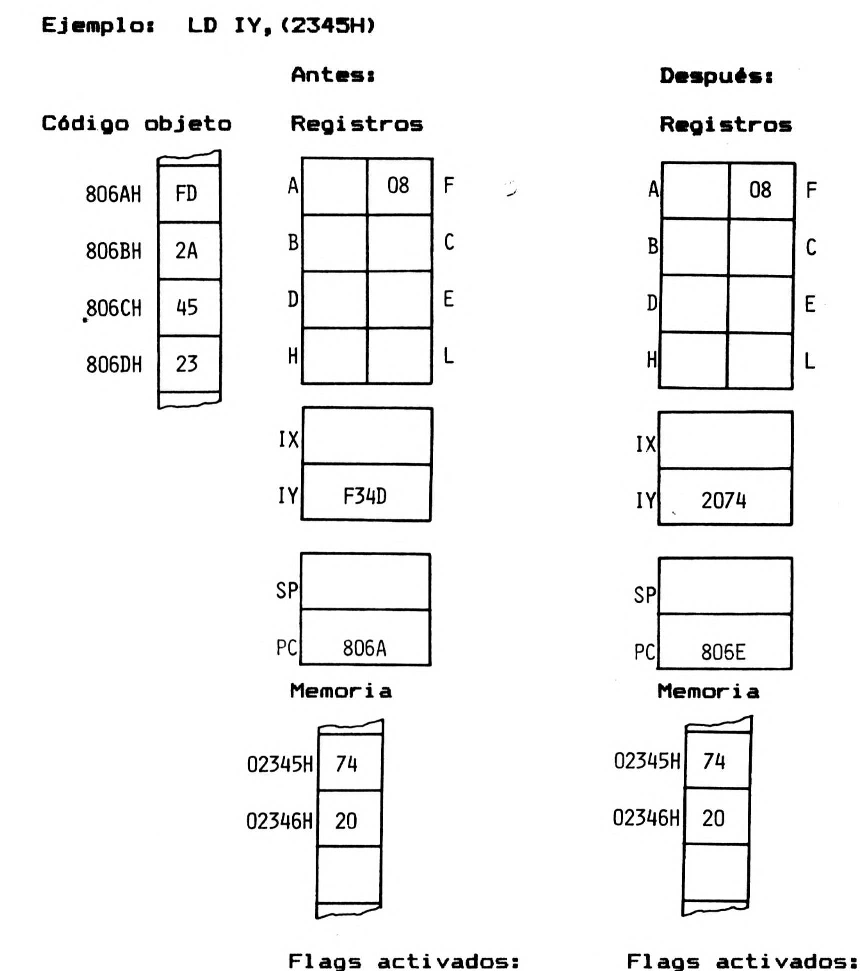
(*) Datos inmediatos de la mitad inf.
Datos inmediatos de la mitad sup.

Ejemplo: LD IY,F34DH



LD IY,(nn)	Carga el registro IY con el contenido de la célula de memoria nn.
Función:	$IY_{inferior}$ <— (nn); $IY_{superior}$<—(nn + 1)
Flags:	S Z - H - P/V N C (ninguna influencia)
Descripción:	El registro de índices IY será cargado con el operando de 16 bits direccionado directamente a través de la dirección nn. Dado que el registro de índices contiene 2 bytes y cada posición de memoria corresponde a 1 byte, se carga en primer lugar la posición de memoria direccionada por nn en la posición inferior de IY. Seguidamente, el contador de programa se incrementa y se carga la posición de memoria de dirección (nn + 1) en la porción superior del par de registros IY.
Ejecución:	Tiempo de ejecución : 5,00 µs Frecuencia de reloj : 4,00 MHz Estados de reloj : 20 Ciclos de máquina : 6
Direccionamiento:	Directo
Formato:	Byte 1) 1 1 1 1 1 1 0 1 FDH Byte 2) 0 0 1 0 1 0 1 0 2AH Byte 3) -----------n----------Porción infer. direc. Byte 4) -----------n----------Porción super. direc.

Ejemplo: LD IY,(2345H)



Flags activados: Flags activados:

LD (IY + d),n — Carga la célula de memoria (IY +d) direccionada indexadamente con los datos n inmediatos.

Función: (IY + d) <—— n

Flags: S Z - H - P/V N C
(ninguna influencia)

Descripción: Se carga la posición de memoria direccionada a través del par de registros IY y el desplazamiento d con el contenido del registro r.

Ejecución:

Tiempo de ejecución	:	4,75 µs
Frecuencia de reloj	:	4,00 MHz
Estados de reloj	:	19
Ciclos de máquina	:	5

Direccionamiento: Indexado/inmediato

Formato:

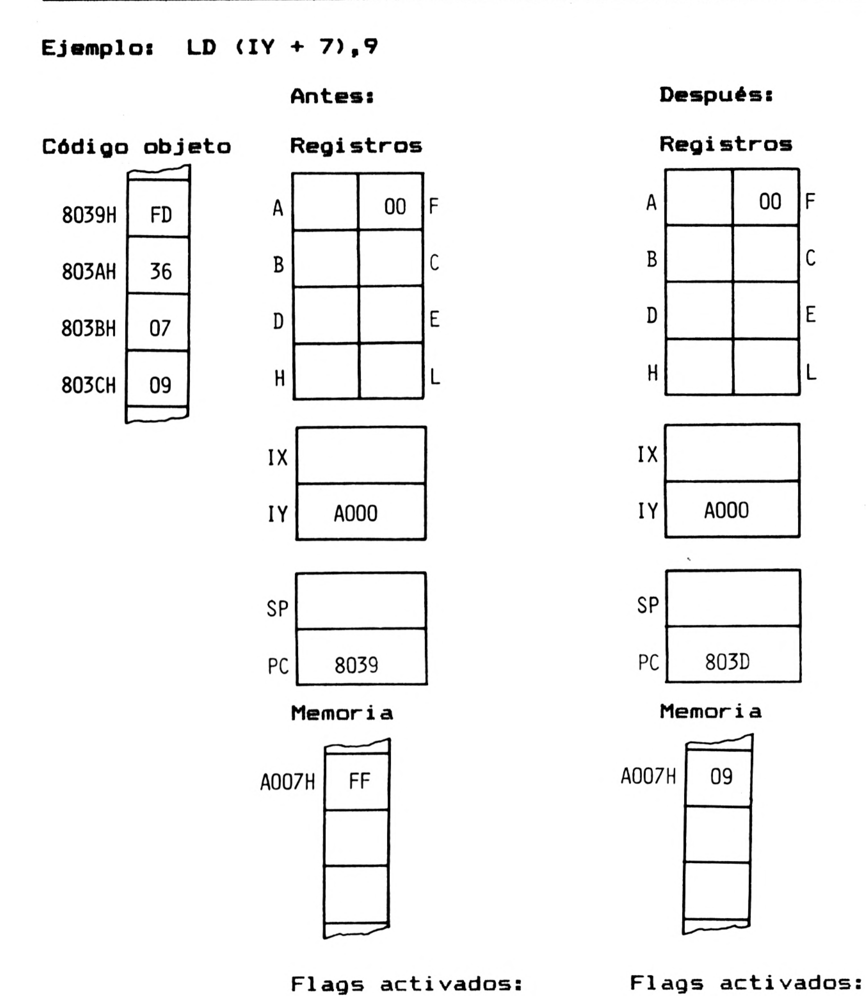
Byte 1) 1 1 1 1 1 1 0 1 FDH

Byte 2) 0 0 1 1 0 1 1 0 36H

Byte 3) ----------d-----------Offset

Byte 4) ----------n-----------Datos inmed.

Ejemplo: LD (IY + 7),9



Flags activados: Flags activados:

LD (IY + d),r — Carga la célula de memoria (IY + d) direccionada indexadamente con el contenido del registro r.

Función: (IY +d) <—— r

Flags: S Z - H - P/V N C
(ninguna influencia)

Descripción: La célula de memoria (IY + d) direccionada indexadamente a través del par de registros IY, además del desplazamiento d, será cargada con el contenido del registro r.

Ejecución:

Tiempo de ejecución	: 4,75 µs
Frecuencia de reloj	: 4,00 MHz
Estados de reloj	: 19
Ciclos de máquina	: 5

Direccionamiento: Indexado

Formato:

Byte 1) 1 1 1 1 1 1 0 1 FDH

Byte 2) 0 1 1 1 0-----r---

Byte 3) ----------d-----------Offset

r puede ser:

A - 111	E - 011
B - 000	H - 100
C - 001	L - 101
D - 010	

Ejemplo: LD (IY + 2),B

Antes:

Código objeto

Dirección	Contenido
8040H	FD
8041H	70
8042H	02

Registros

A		00	F
B	17		C
D			E
H			L

IX	
IY	A000

SP	
PC	8040

Memoria

Dirección	Contenido
A002H	3C

Flags activados:

Después:

Registros

A		00	F
B	17		C
D			E
H			L

IX	
IY	

SP	
PC	8043

Memoria

Dirección	Contenido
A002H	17

Flags activados:

LD (nn),A	Carga la posición de memoria (nn) direccionada directamente con el contenido del acumulador.
Función:	(nn) <—— A
Flags:	S Z - H - P/V N C (ninguna influencia)
Descripción:	La posición de memoria nn direccionada directamente a través de la dirección nn de 16 bits, será cargada con el contenido del acumulador A.
Ejecución:	Tiempo de ejecución : 3,25 µs Frecuencia de reloj : 4,00 MHz Estados de reloj : 13 Ciclos de máquina : 4
Direccionamiento:	Directo
Formato:	Byte 1) 0 0 1 1 0 0 1 0 32H Byte 2) -----------n-----------Mitad infer. direc. Byte 3) -----------n-----------Mitad super. direc.

Ejemplo: LD (9000H),A

Antes:

Código objeto

8046H	32
8047H	00
8048H	90

Registros

A	CD	00	F
B			C
D			E
H			L

IX	
IY	
SP	
PC	8046

Memoria

9000H	F3

Flags activados:

Después:

Registros

A	CD	00	F
B			C
D			E
H			L

IX	
IY	
SP	
PC	8049

Memoria

9000H	CD

Flags activados:

LD (nn),dd — Carga la posición de memoria nn directamente direccionada con el contenido del par de registros dd.

Función: (nn) <— $dd_{inferior}$; (nn + 1) <— $dd_{superior}$

Flags: S Z - H - P/V N C
(ninguna influencia)

Descripción: La posición de memoria (nn) direccionada directamente a través de la dirección nn de 16 bits se carga con la porción inferior del par de registros utilizados y la posición de memoria (nn + 1) con la porción superior del par de registros dd.

Ejecución:

Tiempo de ejecución : 5,00 µs
Frecuencia de reloj : 4,00 MHz
Estados de reloj : 20
Ciclos de máquina : 6

Direccionamiento: Directo

Formato:

Byte 1) 1 1 1 0 1 1 0 1 EDH

Byte 2) 0 1 d d 0 0 1 1

Byte 3) ----------n-----------Mitad infer. direc.

Byte 4) ----------n-----------Mitad super. direc.

dd puede ser:

BC - 00	HL - 10
DE - 01	SP - 11

Ejemplo: LD (8600H),BC

Antes:				Después:		

Código objeto

Dirección	Código
8049H	ED
804AH	43
804BH	00
804CH	86

Antes:

Registros

A			F
B	F4	C9	C
D			E
H			L

IX	
IY	
SP	
PC	8049

Memoria

Flags activados:

Después:

Registros

A			F
B	F4	C9	C
D			E
H			L

IX	
IY	
SP	
PC	804D

Memoria

Flags activados:

<u>LD (nn),IX</u>	**Carga las posiciones de memoria directamente direccionadas mediante nn con el contenido del registro IX.**
Función:	(nn) <— $IX_{inferior}$; (nn+1) <— $I_{xsuperior}$
Flags:	S Z - H - P/V N C (ninguna influencia)
Descripción:	La posición de memoria (nn) direccionada directamente a través de la dirección nn de 16 bits será cargada con la mitad superior del registro de índices IX.
Ejecución:	Tiempo de ejecución : 5,00 µs Frecuencia de reloj : 4,00 MHz Estados de reloj : 20 Ciclos de máquina : 6
Direccionamiento:	Directo
Formato:	Byte 1) 1 1 0 1 1 1 0 1 DDH Byte 2) 0 0 1 0 0 0 1 0 22H Byte 3) -----------n----------Mitad infer. direc. Byte 4) -----------n----------Mitad super. direc.

Ejemplo: LD (9301H),IX

Código objeto

Dirección	Contenido
804DH	DD
804EH	22
804FH	01
8050H	93

Antes:

Registros

A		00	F
B			C
D			E
H			L

IX	6655
IY	
SP	
PC	804D

Memoria

9301H	17
9302H	38

Flags activados:

Después:

Registros

A		00	F
B			C
D			E
H			L

IX	6655
IY	
SP	
PC	8051

Memoria

9301H	55
9302H	66

Flags activados:

LD (nn),IY — **Carga las posiciones de memoria directamente direccionadas mediante nn con el contenido del registro IY.**

Función: (nn) <—— $IY_{inferior}$; (nn+1)<—— $IY_{superior}$

Flags: S Z - H - P/V N C
(ninguna influencia)

Descripción: La posición de memoria (nn) direccionada directamente a través de la dirección nn de 16 bits será cargada mediante la mitad superior del registro de índices IY.

Ejecución:

Tiempo de ejecución	:	5,00 µs
Frecuencia de reloj	:	4,00 MHz
Estados de reloj	:	20
Ciclos de máquina	:	6

Direccionamiento: directo

Formato:

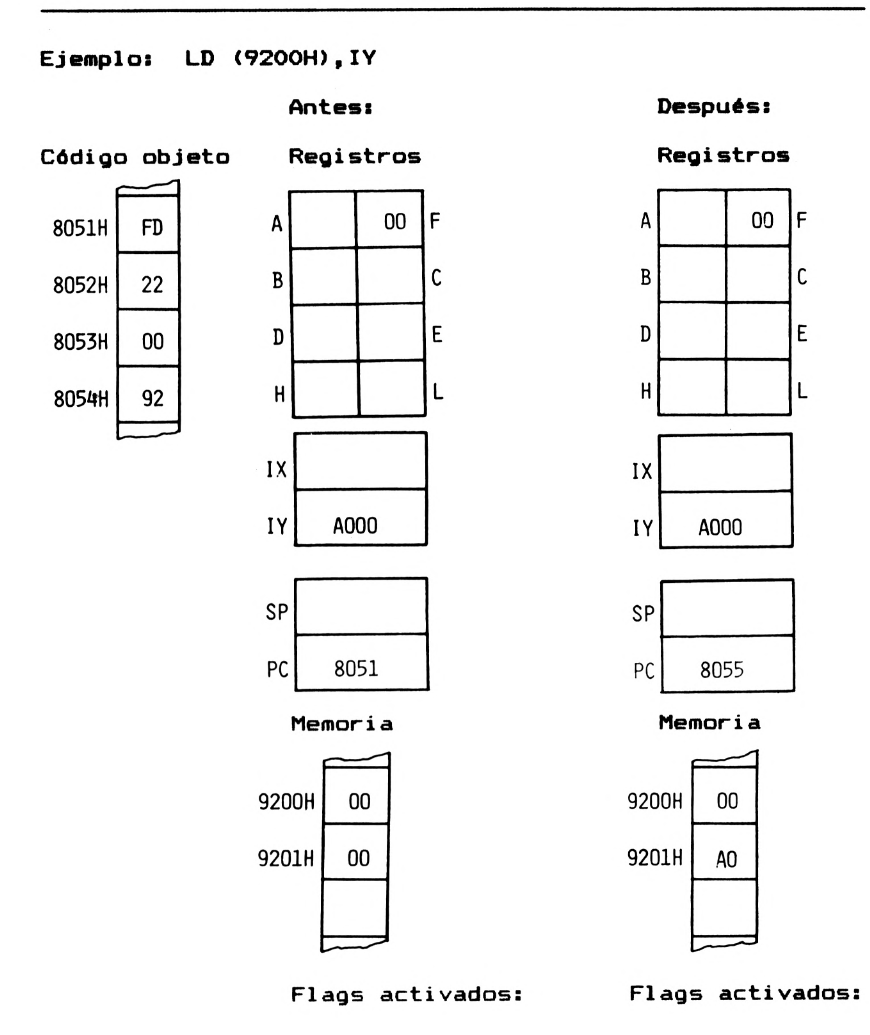
Byte 1) 1 1 1 1 1 1 0 1 FDH

Byte 2) 0 0 1 0 0 0 1 0 22H

Byte 3) ----------n-----------Mitad infer. direc.

Byte 4) ----------n-----------Mitad super. direc.

Ejemplo: LD (9200H),IY



Flags activados: Flags activados:

LD R,A — Carga el registro del Memory-Refresh con el contenido del acumulador.

Función: R <—— A

Flags: S Z - H - P/V N C
(ninguna influencia)

Descripción: Carga el registro R con el contenido del acumulador A. El registro R de refresco contiene la dirección de refresco para la memoria dinámica eventualmente conectada.

Ejecución:

Tiempo de ejecución	:	2,25 µs
Frecuencia de reloj	:	4,00 MHz
Estados de reloj	:	9
Ciclos de máquina	:	2

Direccionamiento: Implícito

Formato:

Byte 1)	1	1	1	0	1	1	0	1	EDH
Byte 2)	0	1	0	0	1	1	1	1	4FH

Ejemplo: LD R,A

Código objeto	
806EH	ED
806FH	4F

Antes:

Registros

A	0D	08	F
B			C
D			E
H			L

IX	
IY	

SP	
PC	806E

Flags activados:

R 18

Después:

Registros

A	0D	08	F
B			C
D			E
H			L

IX	
IY	

SP	
PC	8070

Flags activados:

R 0D

LD r,(HL) — **Carga el registro r con el contenido de la posición de memoria (HL).**

Función: r <—— (HL)

Flags: S Z - H - P/V N C
(ninguna influencia)

Descripción: El registro r será cargado mediante el operando (HL) direcccionado indirectamente a través del par de registros HL.

Ejecución:

Tiempo de ejecución	:	1,75 µs
Frecuencia de reloj	:	4,00 MHz
Estados de reloj	:	7
Ciclos de máquina	:	2

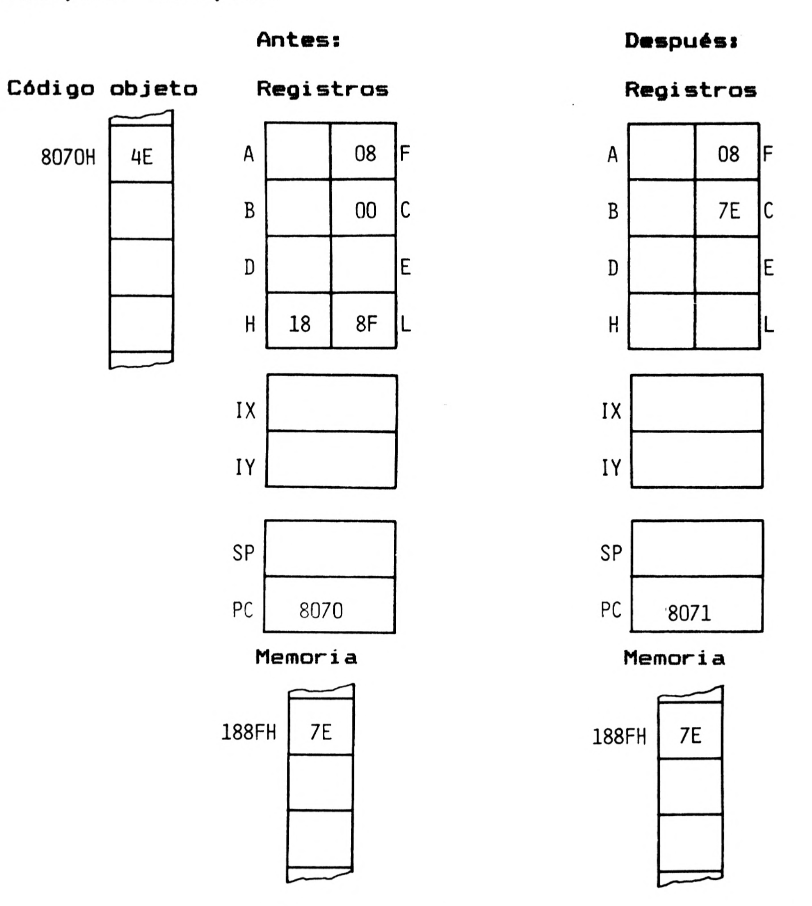
Direccionamiento: Indirecto

Formato: 0 1-----r-----1 1 0

r puede ser:

A - 111	E - 011
B - 000	H - 100
C - 001	L - 101
D - 010	

Ejemplo: LD C,(HL)



Flags activados: Flags activados:

LD r,(IX + d) Carga el registro r con el contenido de la célula de memoria (IX + d) direccionada indexadamente.

Función: r <—— (IX + d)

Flags: S Z - H - P/V N C (ninguna influencia)

Descripción: El registro r será cargado mediante el operando (IX + d) direccionado indexadamente mediante el registro IX y el desplazamiento d.

Ejecución:

Tiempo de ejecución	:	4,75 µs
Frecuencia de reloj	:	4,00 MHz
Estados de reloj	:	19
Ciclos de máquina	:	5

Direccionamiento: Indexado

Formato:

Byte 1) 1 1 0 1 1 1 0 1 DDH

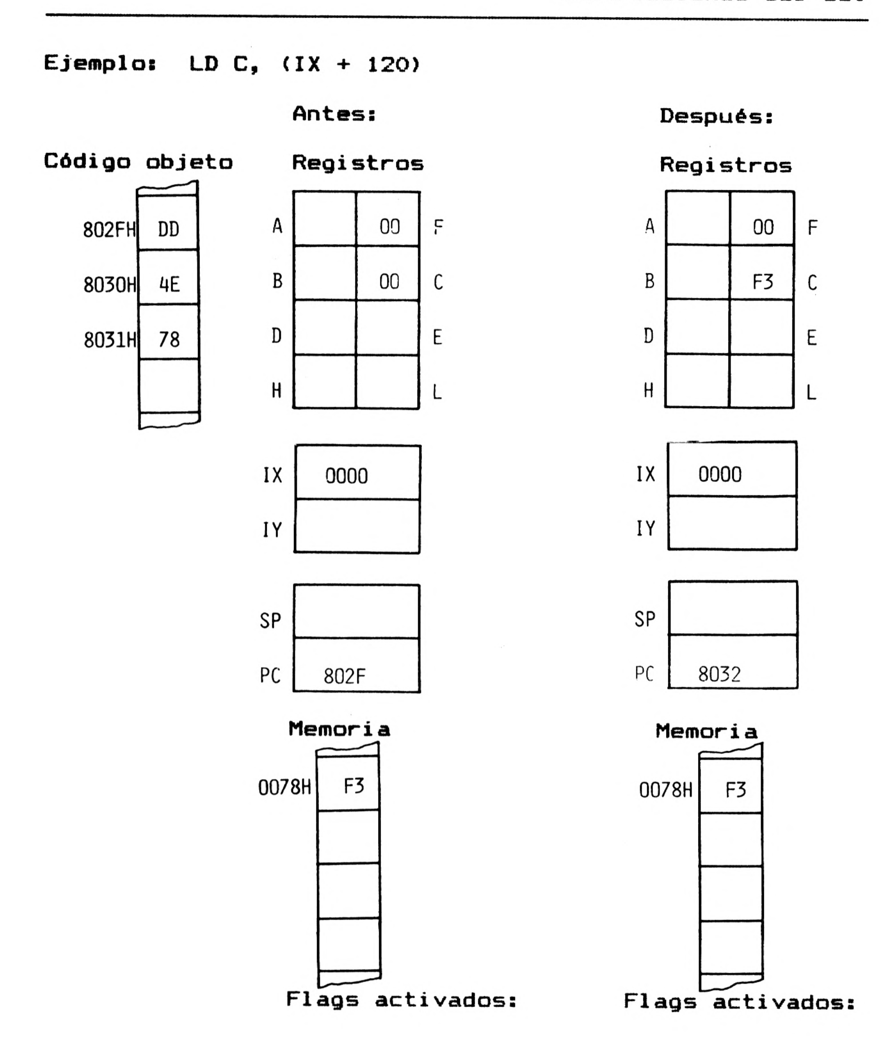
Byte 2) 0 1-----r-----1 1 0

Byte 3) ----------d-----------Offset

r puede ser:

A - 111	E - 011
B - 000	H - 100
C - 001	L - 101
D - 010	

Ejemplo: LD C, (IX + 120)

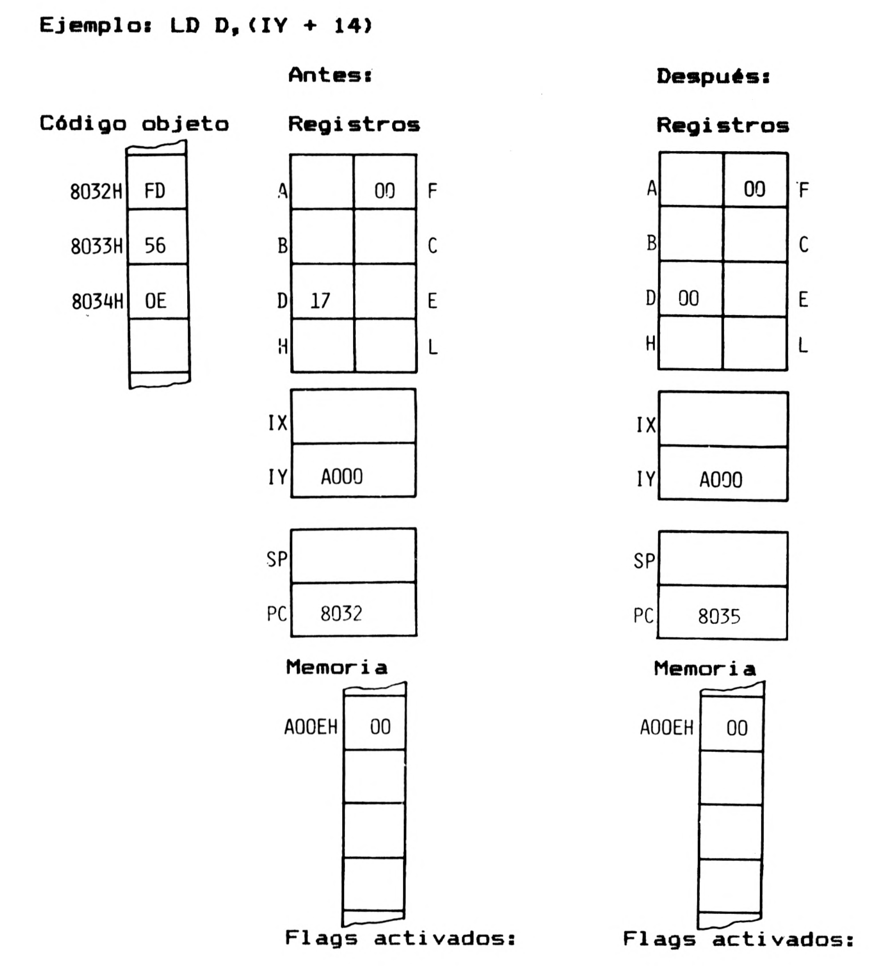


LD r,(IY + d)	Carga el registro r indirecto con el contenido de la célula de memoria (IY + d) direccionada indexadamente.
Función:	r <—— (IY + d)
Flags:	S Z - H - P/V N C (ninguna influencia)
Descripción:	El registro r será cargado con el operando (IY + d) direccionado indexadamente a través del registro IY, y del desplazamiento d.
Ejecución:	Tiempo de ejecución : 4,75 µs Frecuencia de reloj : 4,00 MHz Estados de reloj : 19 Ciclos de máquina : 5
Direccionamiento:	Indexado
Formato:	Byte 1) 1 1 1 1 1 1 0 1 FDH Byte 2) 0 1-----r-----1 1 0 Byte 3) -----------d----------Offset

r puede ser:

A - 111	E - 011
B - 000	H - 100
C - 001	L - 101
D - 010	

Ejemplo: LD D,(IY + 14)



Flags activados: Flags activados:

LD r,n

Carga el registro r con los datos inmediatos n.

Función: <———

Flags: S Z - H - P/V N C
(ninguna influencia)

Descripción: El registro r será cargado con el dato n inmediato de 8 bits, que se encuentra inmediatamente después de la parte operacional de la instrucción.

Ejecución:

Tiempo de ejecución	: 1,75 μs
Frecuencia de reloj	: 4,00 MHz
Estados de reloj	: 7
Ciclos de máquina	: 2

Direccionamiento: Inmediato

Formato:

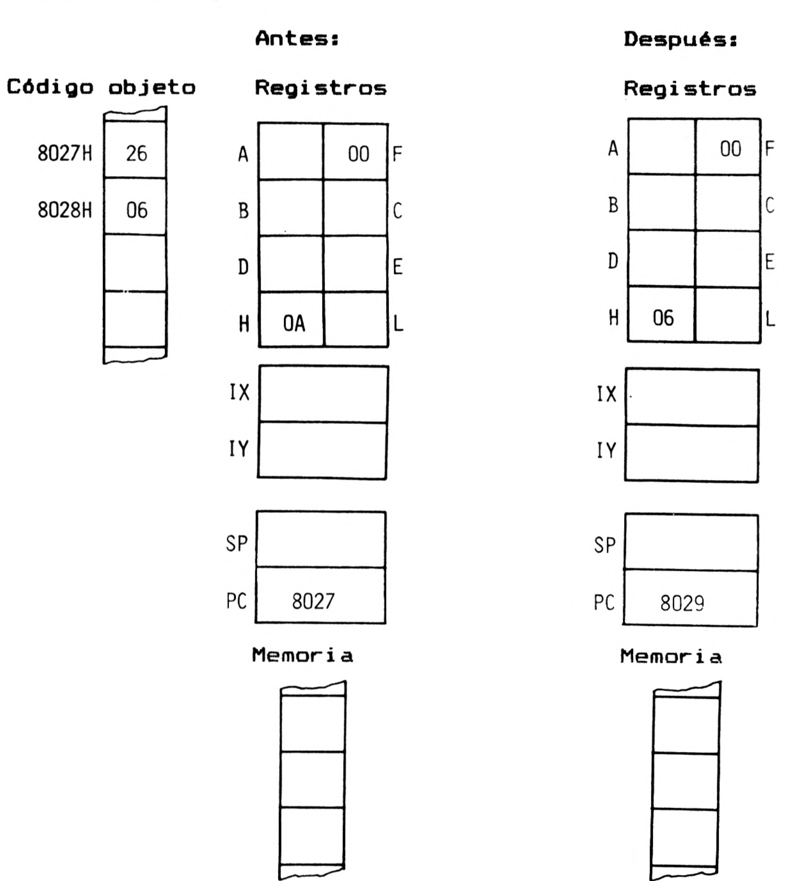
Byte 1) 0 0-----r-----1 1 0

Byte 2) ----------n-----------Datos inmed.

r puede ser:

A - 111	E - 011
B - 000	H - 100
C - 001	L - 101
D - 010	

Ejemplo: LD H,6



Flags activados: Flags activados:

LDr,r'

Carga el registro r con el contenido del registro r'.

Función: r <—— r'

Flags: S Z - H - P/V N C
(ninguna influencia)

Descripción: El registro r será cargado con el contenido del registro r'.

Ejecución:

Tiempo de ejecución	:	1,00 µs
Frecuencia de reloj	:	4,00 MHz
Estados de reloj	:	4
Ciclos de máquina	:	1

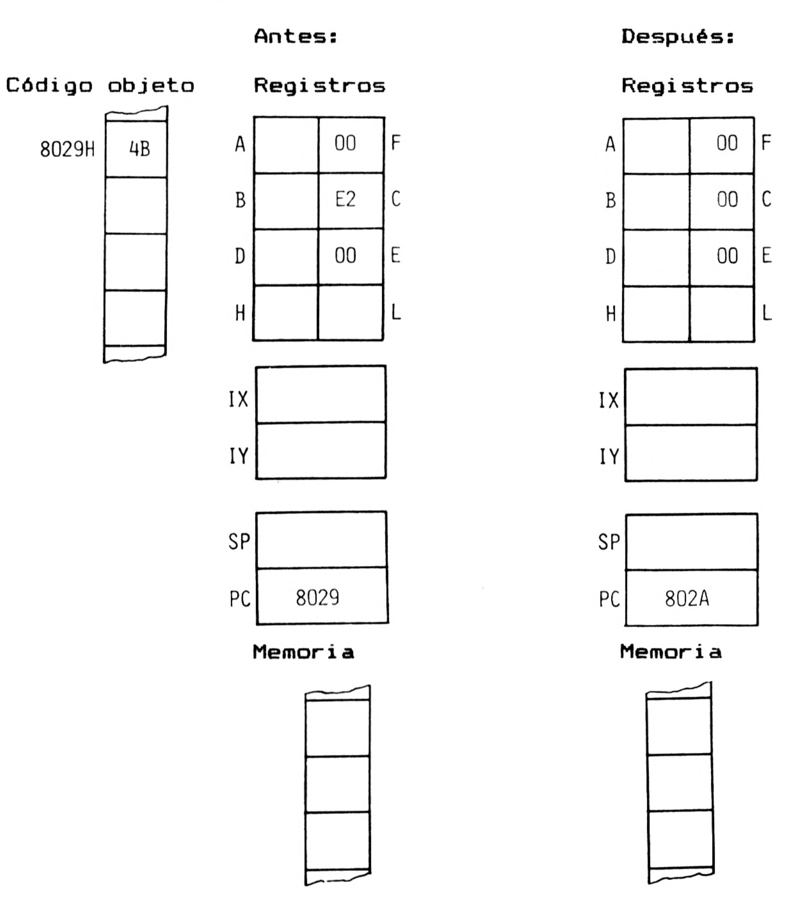
Direccionamiento: Implícito

Formato: 0 1-----r---------r'----

r y r' pueden ser:

A - 111
B - 000
C - 001
D - 010
E - 011
H - 100
L - 101

Ejemplo: LD C,E

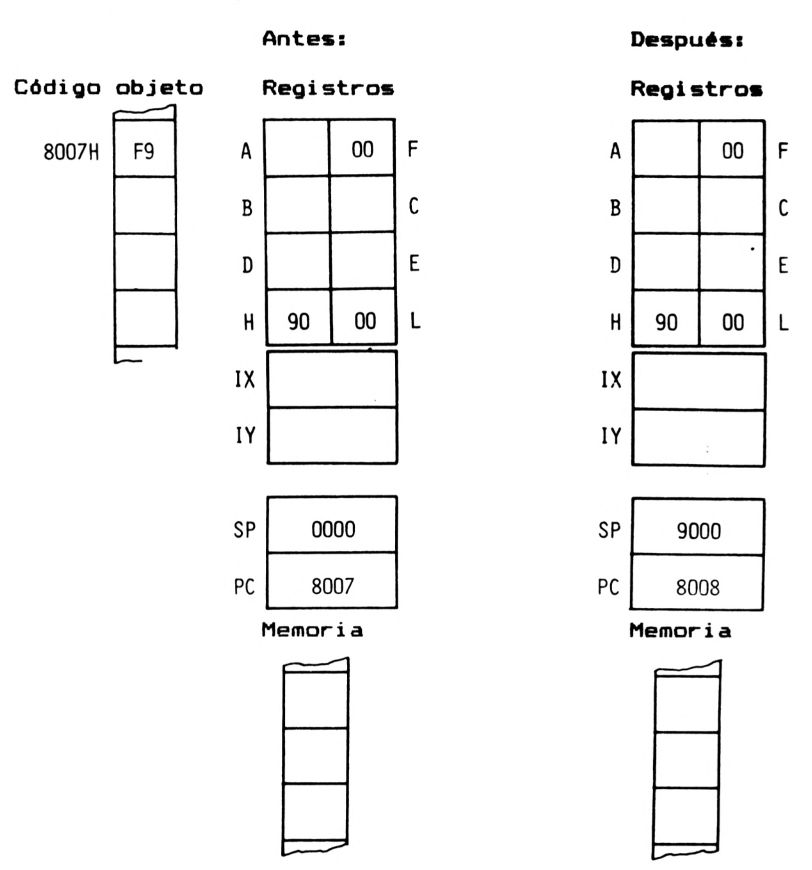


Flags activados:

Flags activados:

LD SP,HL	**Carga el apuntador de pila con el del par de registros HL.**
Función:	SP <—— HL
Flags:	S Z - H - P/V N C (niguna influencia)
Descripción:	El contenido del apuntador de pila SP será igualado a través de la instrucción con el contenido del par de registros HL, es decir SP se carga con HL. El contenido del registro HL quedará invariable.
Ejecución:	Tiempo de ejecución : 1,50 µs Frecuencia de reloj : 4,00 MHz Estados de reloj : 6 Ciclos de máquina : 1
Direccionamiento:	Implícito
Formato:	1 1 1 1 1 0 0 1 F9H

Ejemplo: LD SP,HL



Flags activados: Flags activados:

LD SP,IX — **Carga el apuntador de pila con el contenido del par de registros IX.**

Función: SP <—— IX

Flags: S Z - H - P/V N C
(ninguna influencia)

Descripción: El contenido del apuntador de pila SP será igualado a través de la instrucción con el contenido del par de registros IX, cargando SP con IX. El contenido del registro IX quedará invariable.

Ejecución:

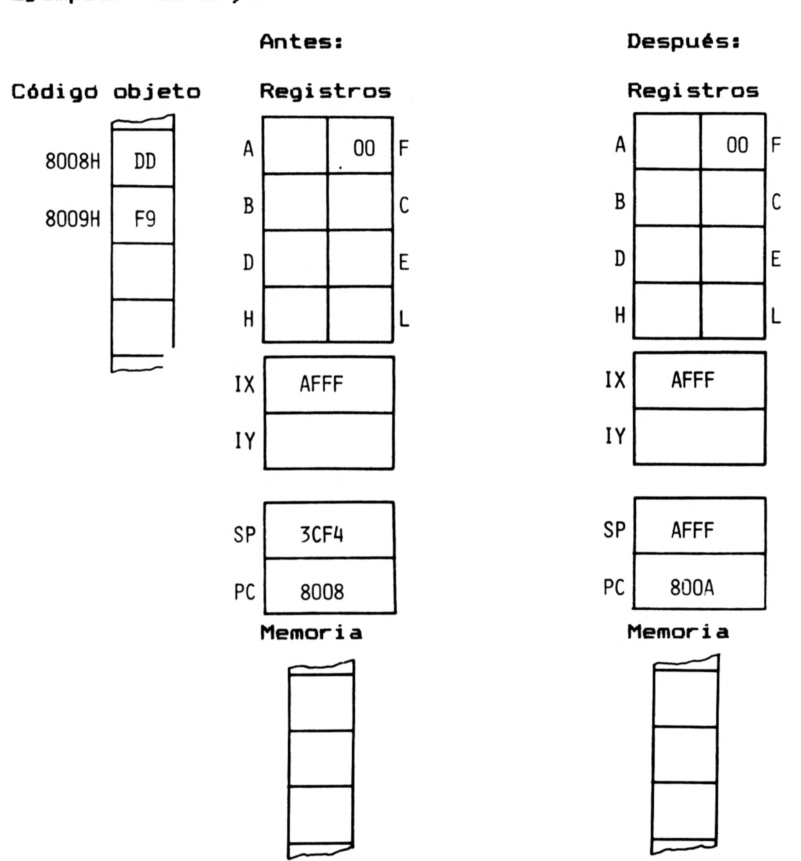
Tiempo de ejecución	:	2,50 µs
Frecuencia de reloj	:	4,00 MHz
Estados de reloj	:	10
Ciclos de máquina	:	2

Direccionamiento: Implícito

Formato:

Byte 1)	1	1	0	1	1	1	0	1	DDH
Byte 2)	1	1	1	1	1	0	0	1	F9H

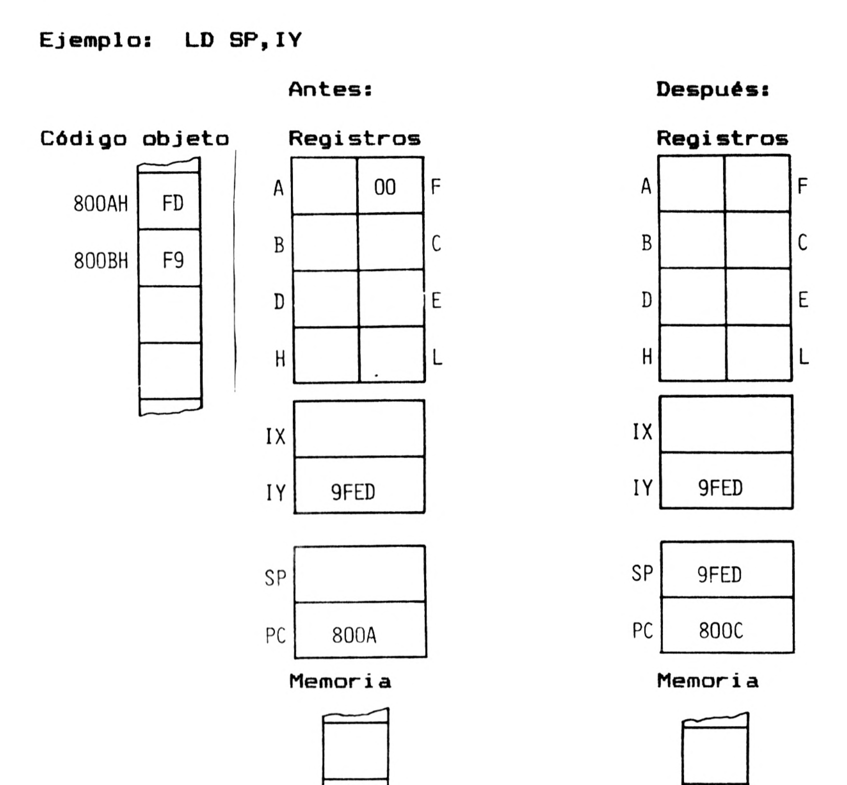
Ejemplo: LD SP,IX



Flags activados: Flags activados:

<u>LD SP,IY</u>	Carga el apuntador de pila con el contenido del par de registros IY.
Función:	SP <—— IY
Flags:	S Z - H - P/V N C (ninguna influencia)
Descripción:	El contenido del apuntador de pila SP será igualado a través de la instrucción con el contenido del par de registros IY, cargando IY en SP. El contenido del registro IY quedará invariable.
Ejecución:	Tiempo de ejecución : 2,50 µs Frecuencia de reloj : 4,00 MHz Estados de reloj : 10 Ciclos de máquina : 2
Direccionamiento:	Implícito
Formato:	Byte 1) 1 1 1 1 1 1 0 1 FDH Byte 2) 1 1 1 1 1 0 0 1 F9H

Ejemplo: LD SP,IY



Flags activados: Flags activados:

LDD Carga de bloques con decremento.

Función: (DE) <— (HL), DE<— DE-1; HL<—HL-1; BC<— BC-1

Flags:

S	Z	-	H	-	P/V	N	C
			0		X	0	

Desactivado, si después de la ejecución BC = 0, en caso contrario activado.

Descripción: La posición de memoria (DE) direccionada indirectamente a través del par de registros DE será cargada con el contenido de la posición de memoria (HL) direccionada indirectamente a través del par de registros HL. Seguidamente los pares de registros DE, HL y BC serán decrementados. El Flag P/V se desactiva únicamente cuando el contenido del par de registros BC sea cero después de la ejecución de la instrucción.

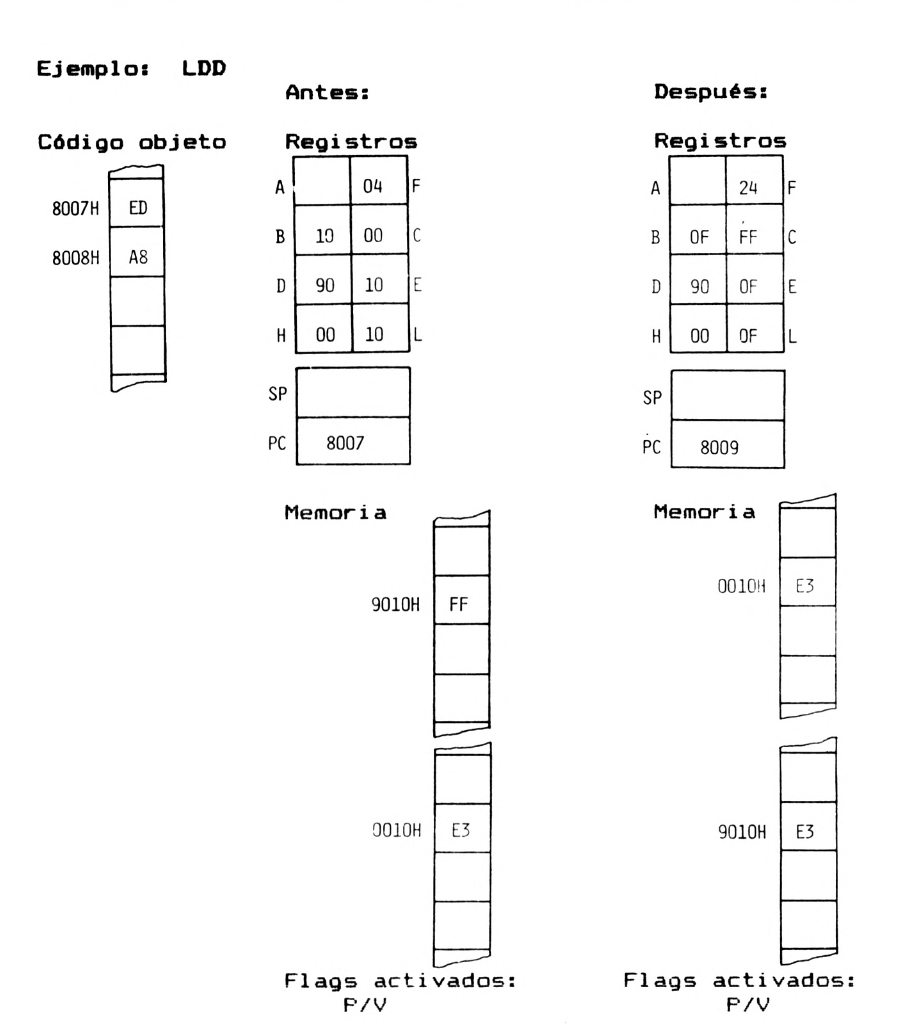
Ejecución:

Tiempo de ejecución : 4,00 µs
Frecuencia de reloj : 4,00 MHz
Estados de reloj : 16
Ciclos de máquina : 4

Direccionamiento: Indirecto

Formato:

Byte 1)	1	1	1	0	1	1	0	1	EDH
Byte 2)	1	0	1	0	1	0	0	0	A8H


Ejemplo: LDD
Antes:
Después:
Código objeto
8007H ED
8008H A8
Registros
A 04 F
B 10 00 C
D 90 10 E
H 00 10 L
SP
PC 8007
Registros
A 24 F
B 0F FF C
D 90 0F E
H 00 0F L
SP
PC 8009
Memoria
9010H FF
0010H E3
Memoria
0010H E3
9010H E3
Flags activados:
P/V
Flags activados:
P/V

LDDR: Carga de bloques con decremento y repetición.

Función: (DE) -- (HL); DE -- DE - 1; HL -- HL - 1; BC -- BC - 1; repite hasta BC=0.

Flags:

S	Z	-	H	-	P/V	N	C
			0		0	0	

Descripción: La posición de memoria (DE) direccionada indirectamente a través del par de registros DE será cargada con el contenido de la posición de memoria (HL) direccionada indirectamente a través del par de registros HL. Seguidamente los pares de registro DE, HL y BC serán decrementados. El flag P/V se desactivará, cuando el contenido del par de registros BC sea cero después de la ejecución de la instrucción. Sólamente cuando el par de registros BC sea diferente de cero, el contador del programa será decrementado en 2, ejecutando así nuevamente la instrucción.

Ejecución: BC $\neq$ 0: 5 ciclos M; 21 estados T; 5,25 µs 4,00 MHz
BC = 0: 4 ciclos M; 16 estados T; 4,00 µs 4,00 MHz.

Direccionamiento: Indirecto

Formato:

Byte 1)	1	1	1	0	1	1	0	1	EDH
Byte 2)	1	0	1	1	1	0	0	0	B8H

Ejemplo: LDDR

Antes:

Código objeto

Dirección	Contenido
8009H	ED
800AH	B8

Registros

A		00	F
B	00	03	C
D	90	0F	E
H	00	10	L

SP	
PC	8009

Memoria

Dirección	Contenido
000DH	F3
000EH	AF
000FH	21
0010H	00
900CH	FF
900DH	FF
900EH	FF
900FH	FF

Flags activados:

Después:

Registros

A		26	F
B	00	00	C
D	90	0C	E
H	00	0D	L

SP	
PC	800BH

Memoria

Dirección	Contenido
000DH	F3
000EH	AF
000FH	21
0010H	00
900CH	F3
900DH	AF
900EH	21
900FH	00

Flags activados:

LDI — **Carga de bloques con incremento.**

Función: (DE) <— (HL); DE <— DE+1; HL <— HL + 1; BC <— BC - 1

Flags:

S	Z	-	H	-	P/V	N	C
			0		X	0	

Si BC = 0 después de la ejecución desactivado, de lo contrario activado.

Descripción: La posición de memoria (DE) direccionada indirectamente a través del par de registros DE será cargada con el contenido de la posición de memoria (HL) direccionada directamente a través del par de registros HL. Seguidamente se incrementan los pares de registros DE y HL y se decrementa el par de registros BC. El flag P/V se desactiva únicamente cuando el contenido del par de registros BC sea cero después de la ejecución de la instrucción.

Ejecución:

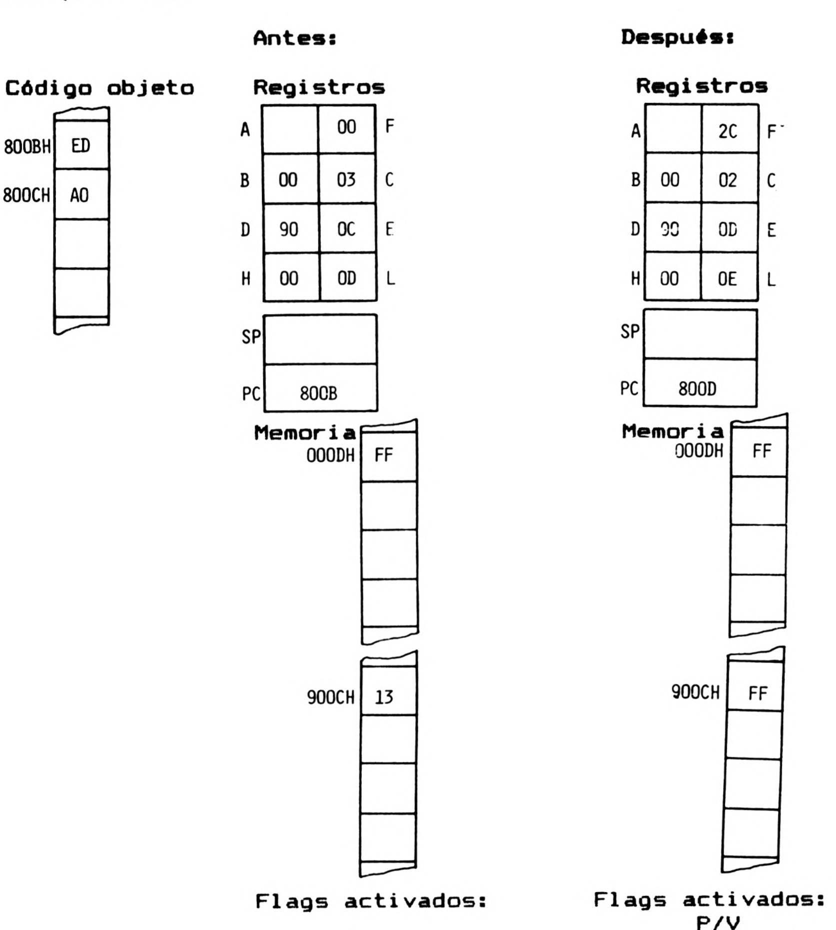
Tiempo de ejecución	: 4,00 µs
Frecuencia de reloj	: 4,00 MHz
Estados de reloj	: 16
Ciclos de máquina	: 4

Direccionamiento: Indirecto

Formato:

Byte 1)	1	1	1	0	1	1	0	1	EDH
Byte 2)	1	0	1	0	0	0	0	0	A0H

Ejemplo: LDI


Antes:
Después:
Código objeto
800BH ED
800CH A0
Registros
A 00 F
B 00 03 C
D 90 0C E
H 00 0D L
SP
PC 800B
Memoria
000DH FF
900CH 13
Registros
A 2C F
B 00 02 C
D 0D E
H 00 0E L
SP
PC 800D
Memoria
000DH FF
900CH FF
Flags activados:
Flags activados:
P/V

LDIR — Carga de bloques con incremento y repetición.

Función: (DE) <— (HL); DE <— DE+1; HL <— HL +1; BC <—BC-1; repetir hasta BC = 0.

Flags:

S	Z	-	H	-	P/V	N	C
			0		0	0	

Descripción: La posición de memoria (DE) direccionada indirectamente a través del par de registros DE será cargada con el contenido de la posición de memoria (HL) direccionada indirectamente a través del par de registros HL. Seguidamente se incrementan los pares de registros DE y HL y se decrementa el par de registros BC. El flag P/V se desactiva únicamente cuando el contenido del par de registros BC sea cero después de la ejecución de la instrucción. Sólamente cuando BC sea diferente de cero, el contador del programa será decrementado en 2, ejecutando así nuevamente la instrucción.

Ejecución: Para BC≠0: 5 ciclos M; 21 estados T; 5,25 µs 4,00 MHz
Para BC=0: 5 ciclos M; 16 estados T; 4,00 µs 4,00 MHz

Direccionamiento: Indirecto

Formato:

Byte 1)	1	1	1	0	1	1	0	1	EDH
Byte 2)	1	0	1	1	0	0	0	0	B0H

Ejemplo: LDIR

Código objeto

Dirección	Contenido
800DH	ED
800EH	B0

Antes:

Registros

A		2C	F
B	00	03	C
D	90	0D	E
H	00	0E	L
SP			
PC	800D		

Memoria

Dirección	Contenido
000EH	FF
000FH	FF
0010H	E3
0011H	F5

Dirección	Contenido
900DH	00
900EH	00
900FH	00
9010H	00

Flags activados:
P/V

Después:

Registros

A		20	F
B	00	00	C
D	90	10	E
H	00	11	L
SP			
PC	800F		

Memoria

Dirección	Contenido
000EH	00
000FH	00
0010H	00
0011H	00

Dirección	Contenido
900DH	FF
900EH	FF
900FH	E3
9010H	F5

Flags activados:

<u>NEG</u>	Invierte el acumulador.
Función:	A <—— 0 <—— A
Flags:	S Z - H - P/V N C
	X X X X 1 X
	C será activado, si A es cero antes de la instrucción. P será activado, si A es 80H.
Descripción:	El contenido del acumulador A será convertido en su complemento binario. El resultado será almacenado en el acumulador A.
Ejecución:	Tiempo de ejecución : 2,00 µs Frecuencia de reloj : 4,00 MHz Estados de reloj : 8 Ciclos de máquina : 2
Direccionamiento:	Implícito
Formato:	Byte 1) 1 1 1 0 1 1 0 1 EDH
	Byte 2) 0 1 0 0 0 1 0 0 44H

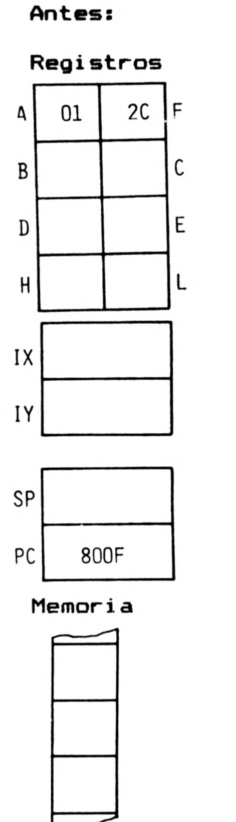
Ejemplo: NEG

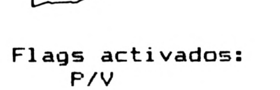
Código objeto

Dirección	Código
800FH	ED
8010H	44

Antes:

Registros





Flags activados:
P/V

Después:

Registros

A	FF	BB	F
B			C
D			E
H			L

IX	
IY	
SP	
PC	8011

Memoria

Flags activados:
S,H,N,C

NOP — **Ninguna operación.**

Función: **Retardo.**

Flags: S Z - H - P/V N C
(ninguna influencia)

Descripción: El procesador no ejecutará ninguna operación; sin embargo emitirá una dirección de refresco. Después de la ejecución de la instrucción, el contador del programa PC será incrementado correspondientemente. En un programa, por ejemplo, se rellenan los espacios vacíos con NOP's.

Ejecución:

Tiempo de ejecución : 1,00 µs
Frecuencia de reloj : 4,00 MHz
Estados de reloj : 4
Ciclos de máquina : 1

Direccionamiento: Implícito

Formato: 0 0 0 0 0 0 0 0 00H

Ejemplo: NOP

	Antes:	Después:
Código objeto	**Registros**	**Registros**
8011H: 0D	A: — F: BB	A: — F: BB
	B: — C: —	B: — C: —
	D: — E: —	D: — E: —
	H: — L: —	H: — L: —
	IX: —	IX: —
	IY: —	IY: —
	SP: —	SP: —
	PC: 8011	PC: 8012
	Memoria	**Memoria**
	Flags activados: S,H,N,C	Flags activados: S,H,N,C

OR s

Relación lógico OR del acumulador y operando s.

Función: A <—— A V s

Flags:

S	Z	-	H	-	P/V	N	C
X	X		0		X	0	0

Descripción: El contenido del acumulador A y del operando definido a través de la s se interrelacionan mediante la relación lógica (OR). La interrelación se efectúa bit a bit y el resultado se almacena en el acumulador.

Ejecución:

s	µs (4,00 MHz)	Estados T	Ciclos M
r	2,00	4	1
n	1,75	7	2
(HL)	1,75	7	2
(IX + d)	4,75	19	5
(IY + d)	4,75	19	5

Direccionamiento: r: implícito/ n: inmediato/ (HL): indirecto/ (IX + d),(IY + d): indexado.

Formato: s: puede ser r,n,(HL),(IX + d), ó (IY + d).

r:		1	0	1	1	0	-----r---			
n:	Byte 1)	1	1	1	1	0	1	1	0	F6H
	Byte 2)	-------------n----------								Datos inmed.
(HL):	Byte 1)	1	0	1	1	0	1	1	0	B6H
(IX+d):	Byte 1)	1	1	0	1	1	1	0	1	DDH
	Byte 2)	1	0	1	1	0	1	1	0	B6H
	Byte 3)	------------d-----------								Offset
(IY+d):	Byte 1)	1	1	1	1	1	1	0	1	FDH
	Byte 2)	1	0	1	1	0	1	1	0	B6H
	Byte 3)	-------------d----------								Offset

r puede ser:

A - 111

B - 000

C - 001

D - 010

E - 011

H - 100

L - 101

Ejemplo: OR (HL)

Antes:

Código objeto

8012H	B6

Registros

A	00	00	F
B			C
D			E
H	00	11	L

IX	
IY	

SP	
PC	8012

Memoria

0011H	F5

Flags activados:

Después:

Registros

A	F5	A4	F
B			C
D			E
H	00	11	L

IX	
IY	

SP	
PC	8013

Memoria

0011H	F5

Flags activados
S,P/V

<u>OTDR</u> Salida de bloques con decremento.

Función: (C) -- (HL); B -- B - 1; HL -- HL -1; repetir hasta que B = 0.

Flags:

S	Z	-	H	-	P/V	N	C
?	1		?		?	1	

Descripción: El port (C) direccionado externamente a través del registro C se carga con el contenido de la posición de memoria (HL) direccionada indirectamente a través del par de registros HL. Seguidamente se decrementan B y HL. El contenido del registro B se compara con cero. Unicamente cuando B sea de cero, se decrementa el contador del programa en 2, ejecutando así nuevamente la instrucción. El contenido del registro C aparece en la mitad inferior del bus de direcciones A0.....A7. B alimenta A8.....A15.

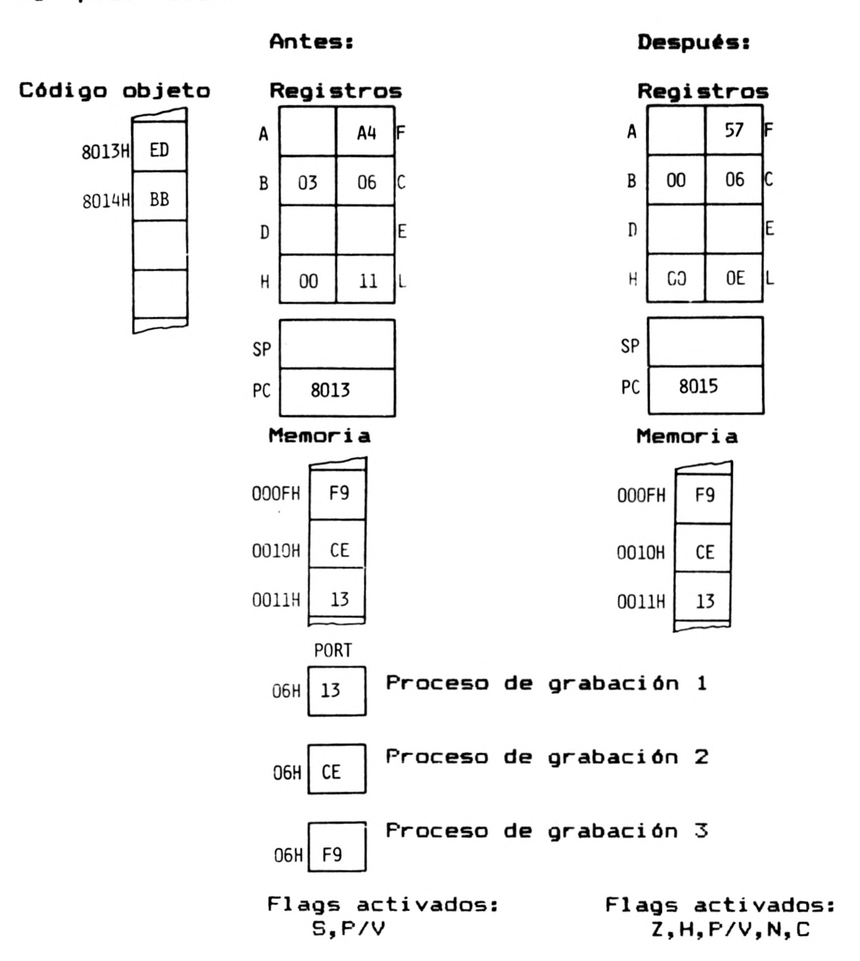
Ejecución: B = 0: 4 ciclos M; 16 estados T; 4,00 µs 4,00 MKz
C ≠ 0: 5 ciclos M; 21 estados T; 5,25 µs 4,00 MHz

Direccionamiento: Externo

Formato:

Byte 1)	1	1	1	0	1	1	0	1	EDH
Byte 2)	1	0	1	1	1	0	1	1	BBH

Ejemplo: OTDR



OTIR — Salida de bloques con incremento.

Función: (C) <— (HL); B <— B-1; HL <— HL + 1; repite hasta que B = 0.

Flags:

S	Z	-	H	-	P/V	N	C
?	1		?		?	1	

Descripción: El Port (C) direccionado externamente a través del registro C se carga con el contenido de la posición de memoria (HL) direccionado indirectamente a través del par de registros HL. Seguidamente se incrementa el par de registros HL y se decrementa el registro B. El contenido del registro B se compara con cero. Unicamente si B es diferente de cero, se decrementará el contador del programa en 2, ejecutando así nuevamente la instrucción. El contenido del registro C aparece en la mitad inferior del bus de direcciones A0.....A7. B alimenta A8..A15.

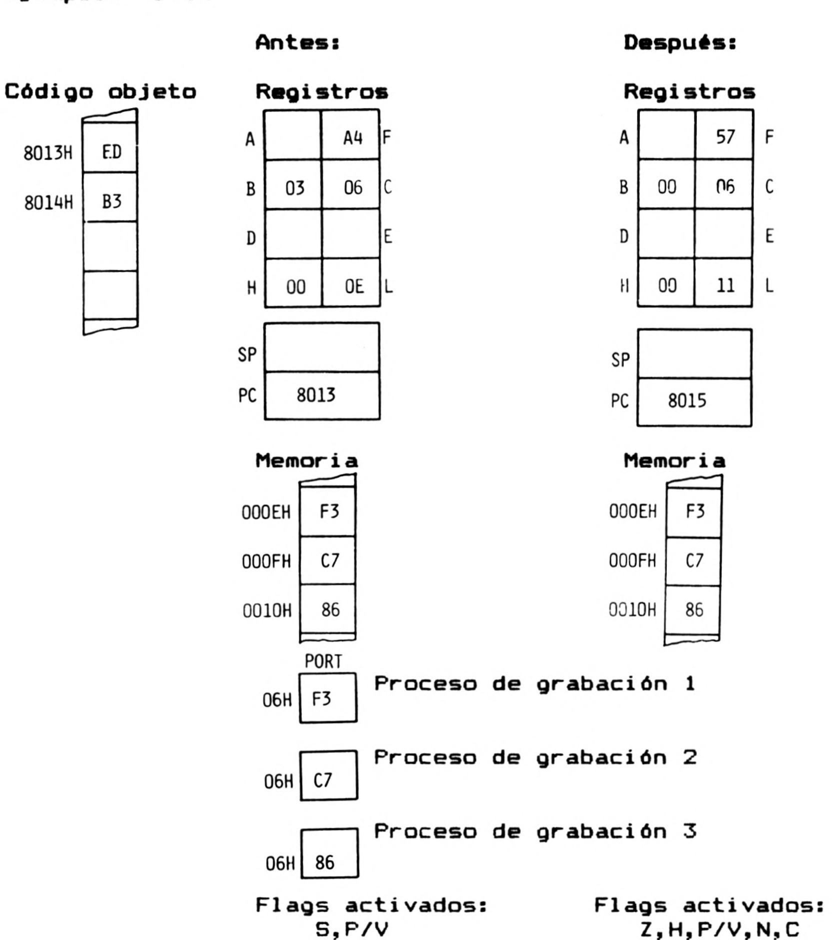
Ejecución: B = 0: 4 ciclos M; 16 estados T; 4,00 µs 4,00 MHz
B ≠ 0: 5 ciclos M; 21 estados T; 5,25 µs 4,00 MHz

Direccionamiento: Externo

Formato:

Byte 1)	1	1	1	0	1	1	0	1	EDH
Byte 2)	1	0	1	1	0	0	1	1	B3H

Ejemplo: OTIR



Proceso de grabación 1

Proceso de grabación 2

Proceso de grabación 3

Flags activados:
S,P/V

Flags activados:
Z,H,P/V,N,C

OUT (C), r — **Salida del registro r al port C.**

Función: (C) <—— r

Flags: S Z - H - P/V N C
(ninguna influencia)

Descripción: El port (C) direccionado externamente a través del registro C se carga con el contenido del registro r. C suministra la dirección a la página cero. En las líneas direccionales A8-A15 aparece el contenido del registro B.

Ejecución:

Tiempo de ejecución	: 3,00 µs
Frecuencia de reloj	: 4,00 MHz
Estados de reloj	: 12
Ciclos de máquina	: 3

Direccionamiento: Externo

Formato:

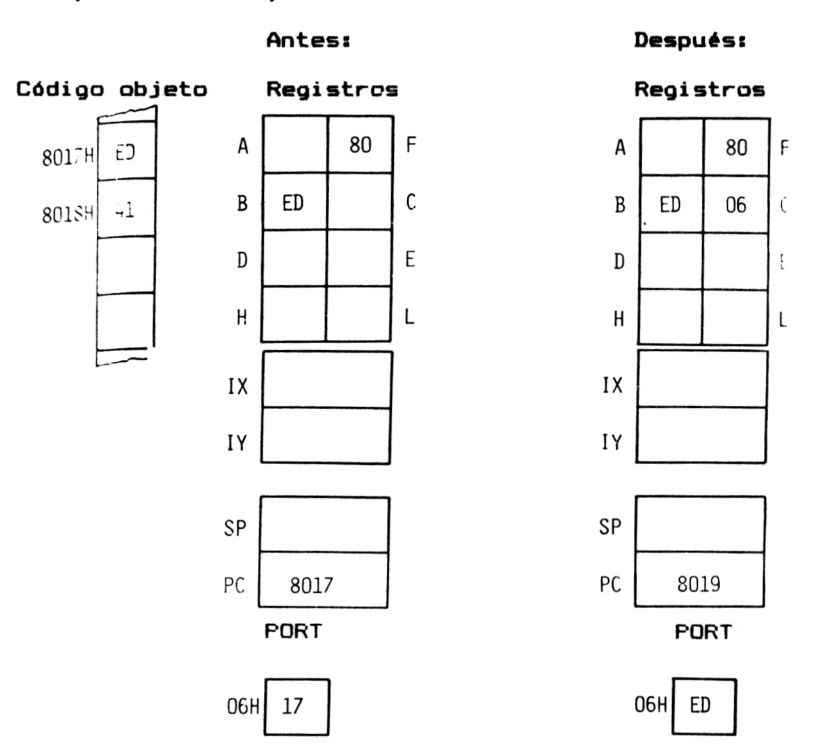
Byte 1) 1 1 1 0 1 1 0 1 EDH

Byte 2) 0 1-----r-----0 0 1

A - 111	E - 011
B - 000	H - 100
C - 001	L - 101
D - 010	

C suministra los bits A0 hasta A7 del bus de direcciones.
B suministra A8 hasta A15 (después de el decremento).

Ejemplo: OUT (C),B



Flags activados:
S

Flags activados:
S

Ejemplo: OUT (6), A

Antes:

Código objeto

8019H	D3
801AH	06

Registros

A	F5	80	F
B			C
D			E
H			L

IX	
IY	

SP	
PC	8019

PORT

06H	16

Flags activados:
S

Después:

Registros

A	F5	80	F
B			C
D			E
H			L

IX	
IY	

SP	
PC	801B

PORT

06H	F5

Flags activados:
S

<u>OUT (N),A</u> — Salida del acumulador al port periférico N.

Función: (N) -- A

Flags: S Z - H - P/V N C
(ninguna influencia)

Descripción: El port (N) direccionado externamente a través del literal N será cargado con el contenido del acumulador A. La N aparecerá en A0 - A7; el contenido del acumulador aparecerá en A8 - A15.

Ejecución:

Tiempo de ejecución	:	2,75 µs
Frecuencia de reloj	:	4,00 MHz
Estados de reloj	:	11
Ciclos de máquina	:	3

Direccionamiento: Externo

Formato:

Byte 1) 1 1 0 1 0 0 1 1 D3H

Byte 2) ----------N----------- Direc. del port

OUTD — Salida con decremento.

Función: (C) <— (HL); B<— B - 1; HL<— HL-1

Flags:

S	Z	-	H	-	P/V	N	C
?	X		?		?	1	

Activado, si B = 0 después de la ejecución, de lo contrario desactivado.

Descripción: El port (C) direccionado externamente a través del registro C será cargado con el contenido de la posición de memoria (HL) direccionada indirectamente a través del par de registros HL. Seguidamente se decrementan B y HL. El contenido del registro B se compara con cero.

Ejecución:

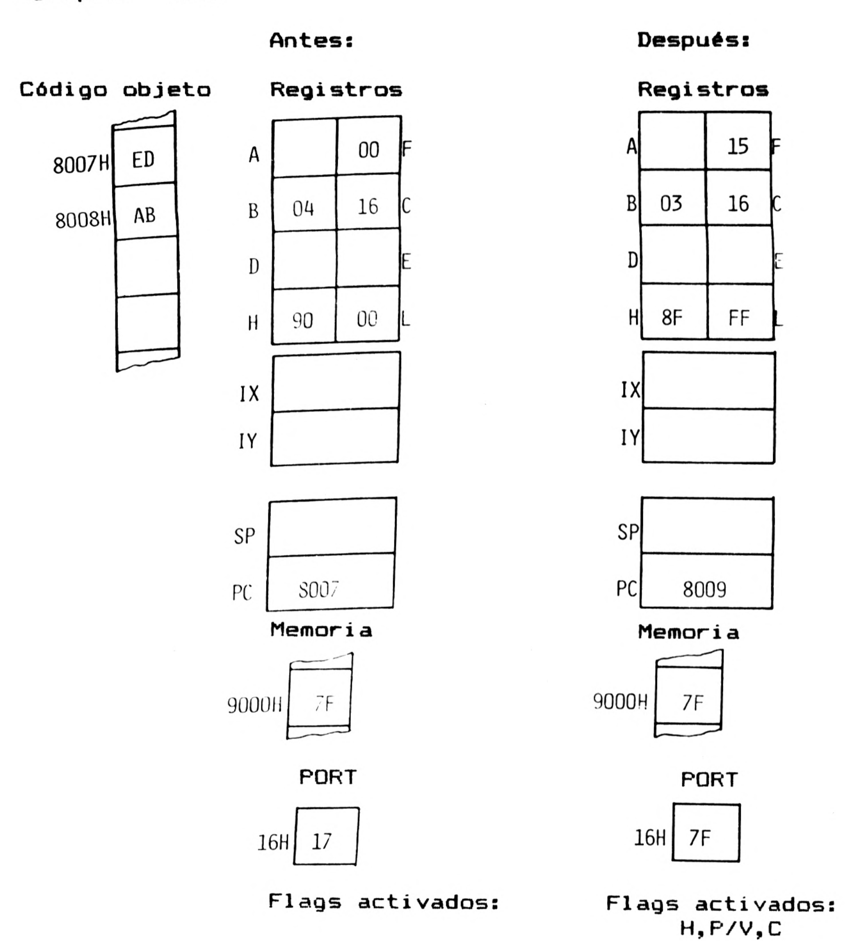
Tiempo de ejecución : 4,00 µs
Frecuencia de reloj : 4,00 MHz
Estados de reloj : 16
Ciclos de máquina : 4

Direccionamiento: Externo

Formato:

Byte 1)	1	1	1	0	1	1	0	1	EDH
Byte 2)	1	0	1	0	1	0	1	1	ABH

Ejemplo: OUTD



OUTI — **Salida con incremento.**

Función: (C) <— (HL); B<— B - 1; HL<— HL +1

Flags:

S	Z	-	H	-	P/V	N	C
?	X		?		?	1	

Activado si B = 0 después de la ejecución, de lo contrario desactivado.

Descripción: El port (C) direccionado externamente a través del registro C será cargado con el contenido de la posición de memoria (HL) direccionada indirectamente a través del par de registros HL. Seguidamente se incrementará el par de registros HL y se decrementará el registro B. El contenido del registro B se compara con cero.

Ejecución:

Tiempo de ejecución : 4,00 µs
Frecuencia de reloj : 4,00 MHz
Estados de reloj : 16
Ciclos de máquina : 4

Direccionamiento: Externo

Formato:

Byte 1)	1	1	1	0	1	1	0	1	EDH
Byte 2)	1	0	1	0	0	0	1	1	A3H

Ejemplo: OUTI

Antes:

Código objeto

8009H	ED
800AH	A3

Registros

A		00	F
B	03	16	C
D			E
H	8F	FF	L

SP	
PC	8009

SP	
PC	8009

Memoria

8FFFH	38

PORT

16H	4C

Flags activados:

Después:

Registros

A		00	F
B	02	16	C
D			E
H	90	00	L

SP	
PC	800BH

SP	
PC	800BH

Memoria

8FFFH	38

PORT

16H	38

Flags activados:

POP IX Extrae el registro IX de la pila.

Función: $IX_{inferior}$ <— (SP); $IX_{superior}$ <—(SP+1);

SP <— SP + 2

Flags: S Z - H - P/V N C
(ninguna influencia)

Descripción: La instrucción cargará los dos últimos elementos de la pila en el registro de índices IX. En primer lugar se cargará el elemento de la pila (SP) direccionado indirecta mente a través del apuntador de la pila SP en la mitad inferior del registro de índices IX y seguidamente se cargará el elemento de la pila (SP + 1) direccionado indirectamente a través del apuntador de la pila SP + 1 en la mitad superior del registro de índices IX. A continuación el apuntador de pila volverá a ser incrementado.

Ejecución:

Tiempo de ejecución	:	3,50 µs
Frecuencia de reloj	:	4,00 MHz
Estados de reloj	:	14
Ciclos de máquina	:	4

Direccionamiento: Indirecto

Formato: Byte 1) 1 1 0 1 1 1 0 1 DDH

Byte 2) 1 1 1 0 0 0 0 1 E1H

Ejemplo: POP IX

Antes:

Código objeto

800BH	DD
800CH	E1

Registros

A		00	F
B			C
D			E
H			L

IX	7C3F
IY	

SP	9000
PC	800B

Memoria

9000H	C4
9001H	70

Flags activados:

Después:

Registros

A		00	F
B			C
D			E
H			L

IX	7064
IY	

SP	9002
PC	800D

Memoria

9000H	C4
9001H	70

Flags activados:

POP IY Extrae el registro IY de la pila.

Función: $IY_{inferior}$ <— (SP); $IY_{superior}$ <—(SP+1);

SP <— SP + 2

Flags: S Z - H - P/V N C
(ninguna influencia)

Descripción: La instrucción cargará los dos últimos elementos de la pila en el registro de índices IY. En primer lugar se cargará el elemento de la pila (SP) direccionado indirectamente a través del apuntador de pila SP en la mitad inferior del registro de índices IY y a continuación se cargará el elemento de la pila (SP + 1) direccionado indirectamente a través del apuntador de la pila SP + 1 en la mitad superior del registro de índices IY. Seguidamente se incrementará de nuevo el apuntador de pila (Stack Pointer).

Ejecución:

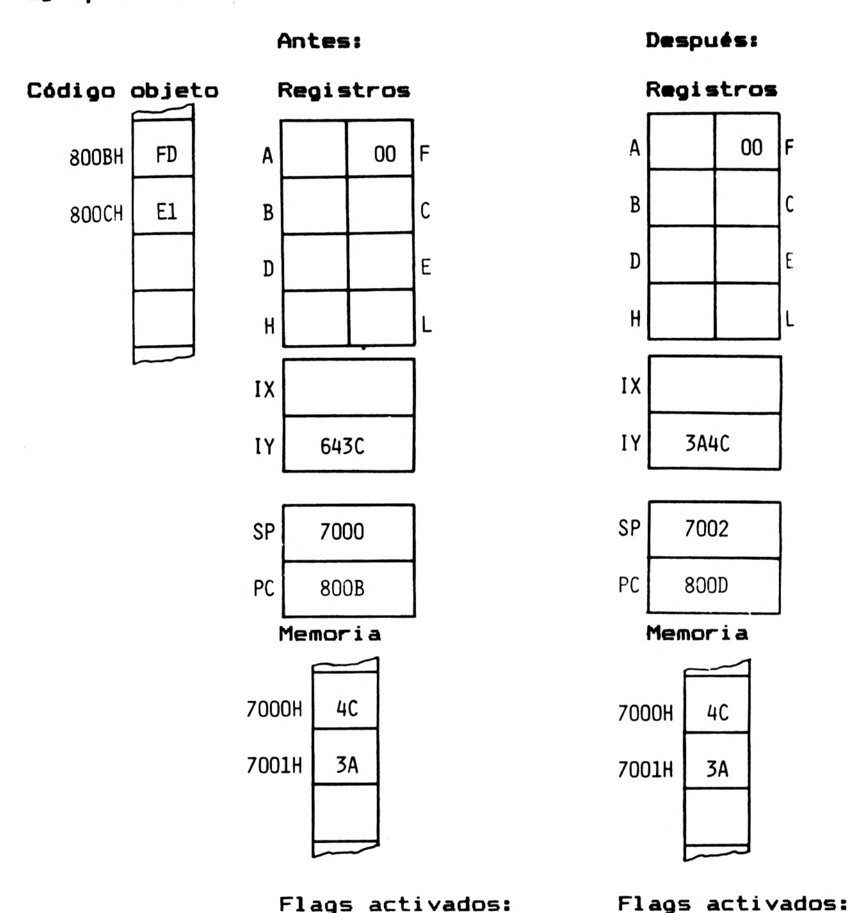
Tiempo de ejecución	:	1,00 µs
Frecuencia de reloj	:	4,00 MHz
Estados de reloj	:	14
Ciclos de máquina	:	4

Direccionamiento: Indirecto

Formato:

Byte 1)	1	1	1	1	1	1	0	1	FDH
Byte 2)	1	1	1	0	0	0	0	1	E1H

Ejemplo: POP IY



Flags activados: Flags activados:

POP qq — Extrae el par de registros qq de la pila (Stack).

Función: $qq_{inferior}$<——(SP); $qq_{superior}$<——(SP + 1);

SP<——SP + 2

Flags: S Z - H - P/V N C
(ninguna influencia)

Descripción: La instrucción carga los dos últimos elementos de la pila en el par de registros qq. En primer lugar se carga el elemento de la pila (SP) direccionado indirectamente a través del apuntador de pila SP en la mitad inferior del par de registros qq y a continuación se cargará el elemento de la pila direccionado indirectamente a través del apuntador de pila SP + 1 en la mitad superior del par de registros qq. Seguidamente volverá a incrementarse el apuntador de pila (Stack Pointer).

Ejecución:

Tiempo de ejecución	:	2,50 µs
Frecuencia de reloj	:	4,00 MHz
Estados de reloj	:	10
Ciclos de máquina	:	3

Direccionamiento: Indirecto

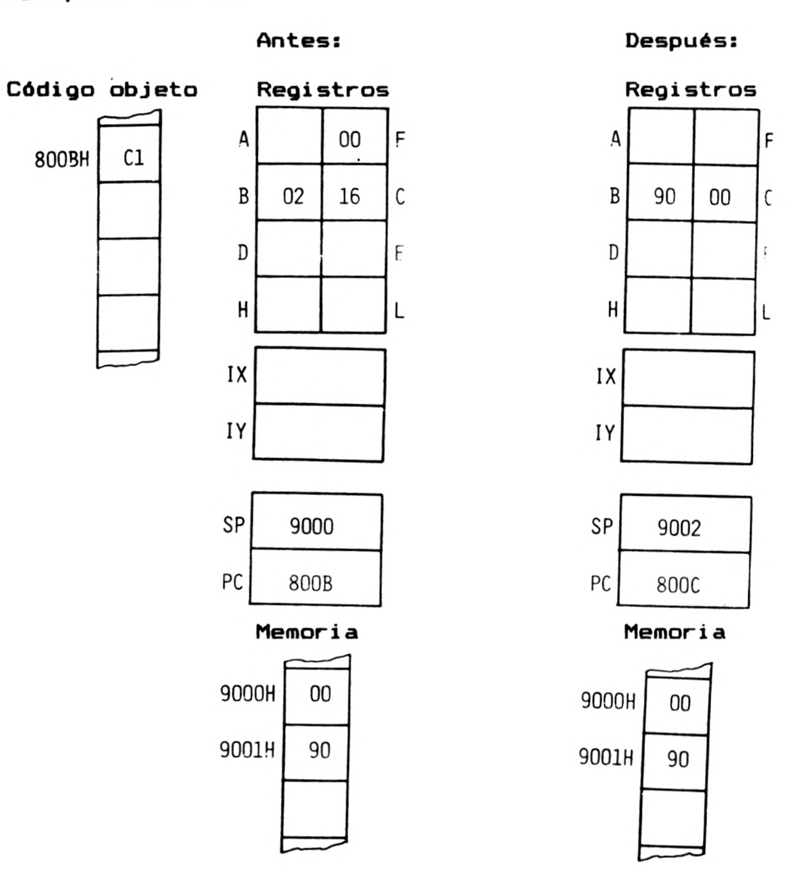
Formato: 1 1 q q 0 0 0 1

qq puede ser:

BC - 00 HL - 10

DE - 01 AF - 11

Ejemplo: POP BC

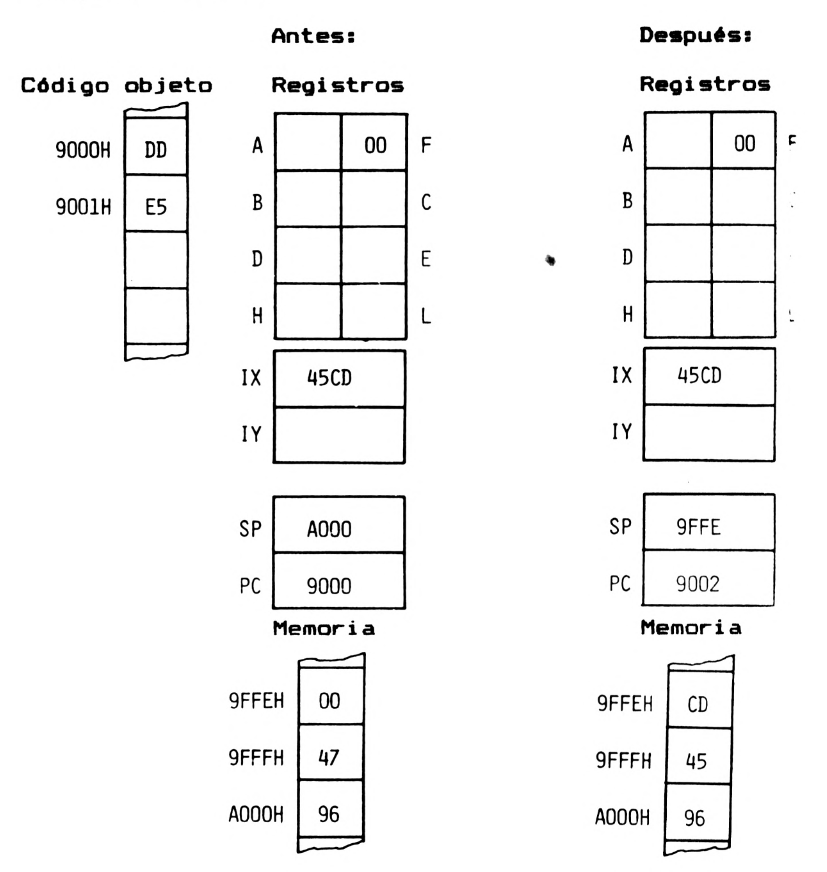


Flags activados:

Flags activados:

<u>PUSH IX</u>	Almacena el registro IX en la pila (Stack).
Función:	$(SP-1) \leftarrow IX_{superior}$; $(SP-2) \leftarrow IX_{inferior}$ $SP \leftarrow SP - 2$
Flags:	S Z - H - P/V N C (ninguna influencia)
Descripción:	La instrucción carga el contenido del registro de índices IX en la pila. Em primer lugar se cargará la célula de la pila (SP - 1) direccionada indirectamente a través del apuntador de pila SP - 1 decrementado con la mitad superior del registro de índices IX y seguidamente se carga la célula de la pila direccionada indirectamente a través del apuntador de pila SP - 2 nuevamente decrementado con la mitad inferior del registro de índices IX.
Ejecución:	Tiempo de ejecución : 3,75 µs Frecuencia de reloj : 4,00 MHz Estados de reloj : 15 Ciclos de máquina : 4
Direccionamiento:	Indirecto
Formato:	Byte 1) 1 1 0 1 1 1 0 1 DDH Byte 2) 1 1 1 0 0 1 0 1 E5H

Ejemplo: PUSH IX



Flags activados: Flags activados:

PUSH IY Almacena el registro IY en la pila.

Función: (SP-1)<—$IY_{superior}$; (SP-2)<— $IY_{inferior}$ SP<—SP - 2

Flags: S Z - H - P/V N C
(ninguna influencia)

Descripción: La instrucción carga el contenido del registro de índices IY en la pila. En primer lugar se carga la célula de la pila (SP - 1) direccio nada indirectamente a través del apuntador de pila SP - 1 decrementado con la mitad superior del registro de índices IY y seguidamente se carga la célula de la pila direccionada indirectamente a través del apuntador de pila (SP - 2) nuevamente decrementado con la mitad inferior del registro de índices IY.

Ejecución:

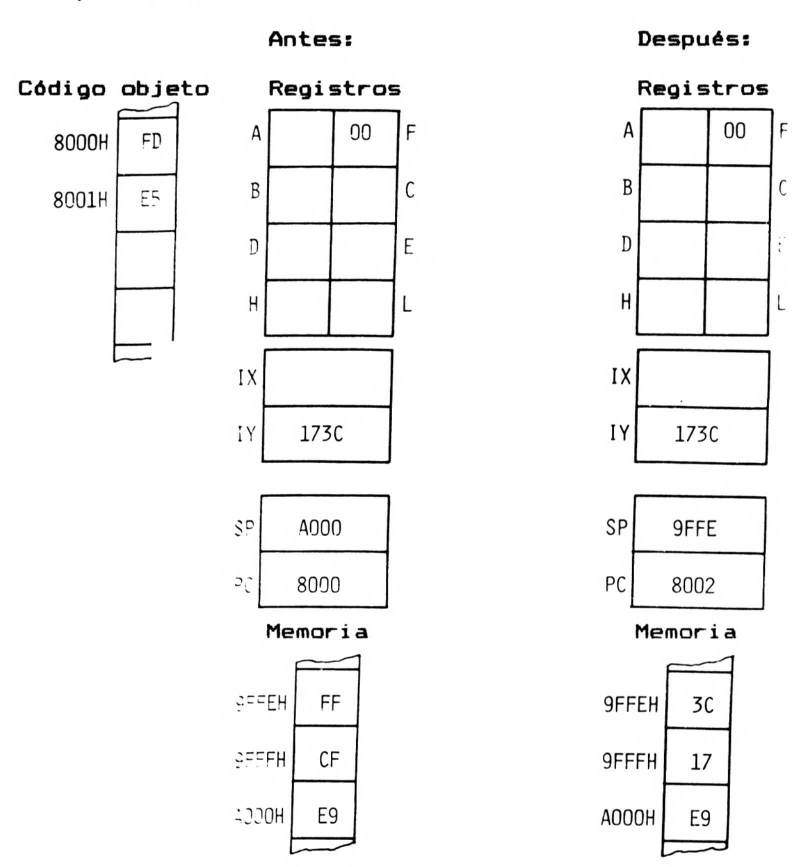
Tiempo de ejecución	:	3,75 µs
Frecuencia de reloj	:	4,00 MHz
Estados de reloj	:	15
Ciclos de máquina	:	3

Direccionamiento: Indirecto

Formato:

Byte 1)	1	1	1	1	1	1	0	1	FDH
Byte 2)	1	1	1	0	0	1	0	1	E5H

Ejemplo: PUSH IY



Flags activados: Flags activados:

PUSH qq

Almacena el par de registros qq en la pila.

Función:

$(SP-1) \leftarrow qq_{superior}$; $(SP-2) \leftarrow qq_{inferior}$ $SP \leftarrow SP - 2$

Flags:

S Z - H - P/V N C
(ninguna influencia)

Descripción:

La instrucción carga el contenido del par de registros qq en la pila. En primer lugar se carga la célula de la pila (SP - 1) direccionada indirectamente a través del apuntador de pila SP - 1 decrementado con la mitad superior del par de registros qq y seguidamente se carga la célula de la pila direccionada indirectamente a través del apuntador de pila SP - 2 nuevamente decrementado con la mitad inferior del par de registros qq.

Ejecución:

Tiempo de ejecución	:	3,25 µs
Frecuencia de reloj	:	4,00 MHz
Estados de reloj	:	11
Ciclos de máquina	:	3

Direccionamiento: Indirecto

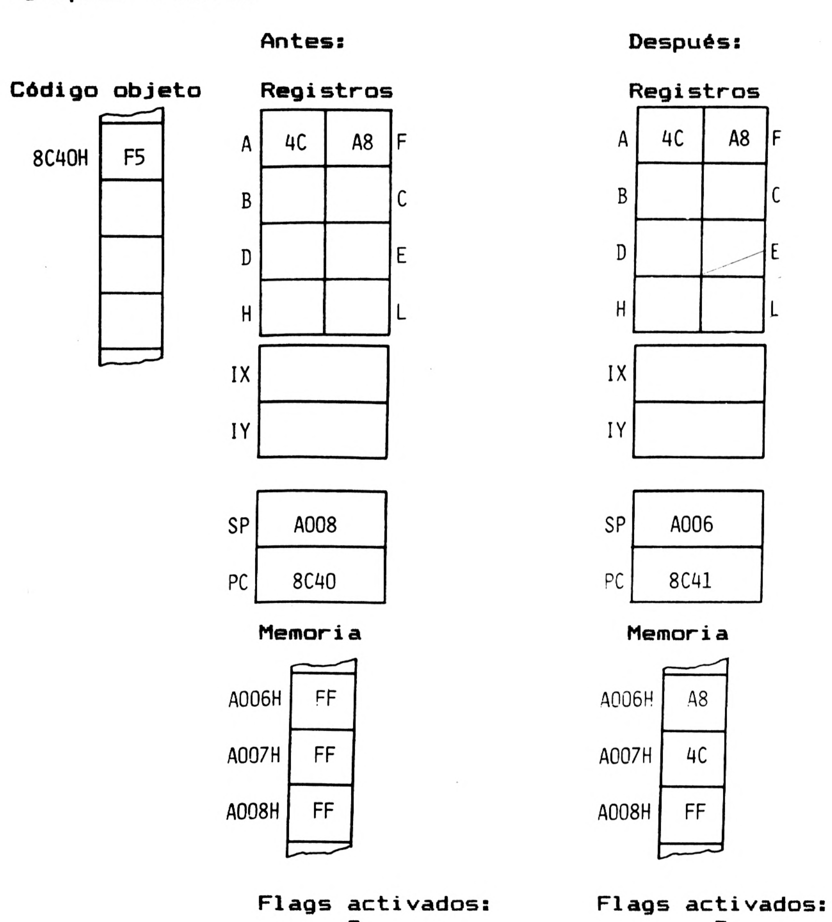
Formato:

1 1 q q 0 1 0 1

qq puede ser:

BC - 00	HL - 10
DE - 01	AF - 11

Ejemplo: PUSH AF



Flags activados: S

Flags activados: S

RES b,s — Desactiva el bit b del operando s.

Función: s_b <— 0

Flags: S Z - H - P/V N C
(ninguna influencia)

Descripción: El bit especificado mediante b del operando activado a través de s se desactiva (valor 0).

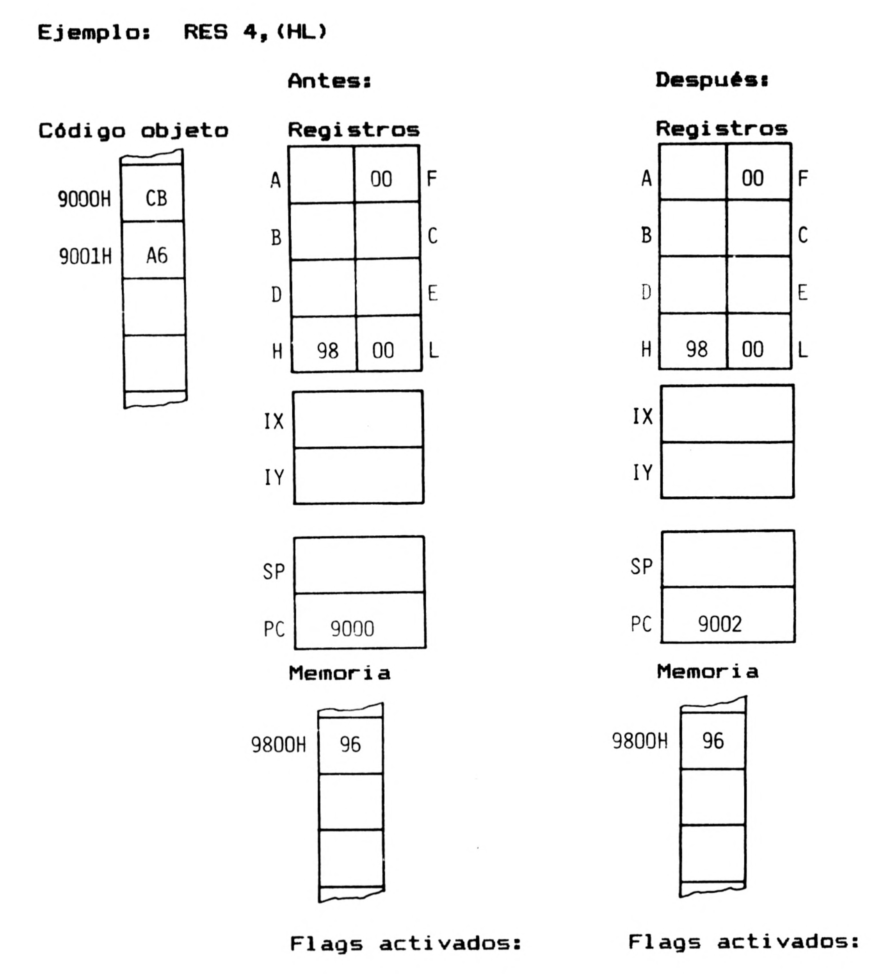
Ejecución:

s	µs(4,00 MHz)	Estados T	Ciclos M
r	2,00	8	2
(HL)	3,75	15	4
(IX + d)	5,75	23	6
(IY + d)	5,75	23	6

Direccionamiento: r: implícito/ (HL):indirecto/ (IX d), (IY d): indexado.

```
Formato: s:
         r:          Byte 1): 1  1  0  0  1  0  1  1  CBH
                     Byte 2): 1  0-----b---------r---
         (HL):       Byte 1):  1  1  0  0  1  0  1  1  CBH
                     Byte 2): 1  0-----b-----1  1  0
         (IX + d):   Byte 1): 1  1  0  1  1  1  0  1  DDH
                     Byte 2): 1  1  0  0  1  0  1  1  CBH
                     Byte 3): -----------d-----------Offset
         (IY + d):   Byte 1): 1  1  1  1  1  1  0  1  FDH
                     Byte 2): 1  1  0  0  1  0  1  1  CBH
                     Byte 3): -----------d-----------Offset
                     Byte 4): 1  0-----b-----1  1  0
         b puede ser:
                        0 - 000          4 - 100
                        1 - 001          5 - 101
                        2 - 010          6 - 110
                        3 - 011          7 - 111
         r puede ser:
                        A - 111          E - 011
                        B - 000          H - 100
                        C - 001          L - 101
                        D - 010
```

Ejemplo: RES 4,(HL)



Flags activados: Flags activados:

RET — **Retorno del subprograma.**

Función: $PC_{inferior}$ <— (SP); $PC_{superior}$ <— (SP + 1);

SP <— SP + 2

Flags: S Z - H - P/V N C
(ninguna influencia)

Descripción: El contador del programa PC se carga con las dos últimas inserciones de la pila (para más detalles véase la instrucción "POP"). De esta manera el programa podrá continuar con la siguiente instrucción del programa principal.

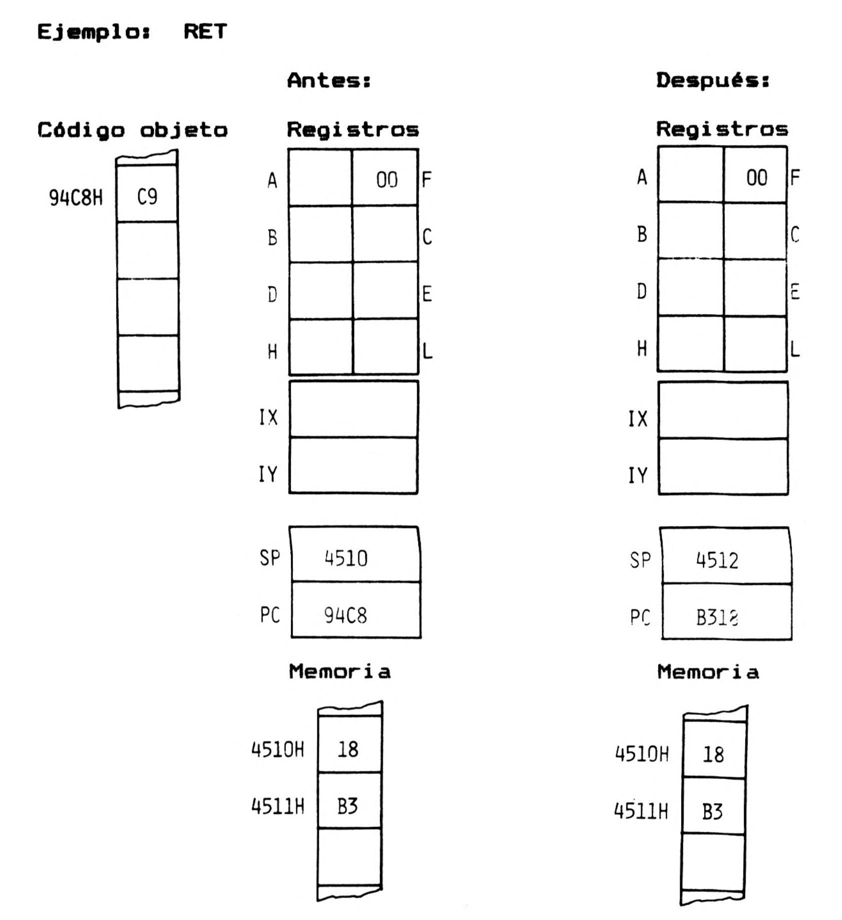
Ejecución:

Tiempo de ejecución	:	2,50 µs
Frecuencia de reloj	:	4,00 MHz
Estados de reloj	:	10
Ciclos de máquina	:	3

Direccionamiento: Indirecto

Formato: 1 1 0 0 1 0 0 1 C9

Ejemplo: RET



Flags activados:

Flags activados:

RET cc — Retorno condicionado del subprograma.

Función: Cuando se cumple cc, entonces $PC_{inferior}$ <—(SP); $PC_{superior}$ <— (SP + 1); SP <— SP + 2

Flags: S Z - H - P/V N C
(ninguna influencia)

Descripción: El contador del programa PC se carga con las dos últimas inserciones de la pila (para más detalles véase la instrucción "POP"). De esta manera el programa podrá continuar con la siguiente instrucción del programa principal. Si la condición cc se cumple, se efectúa el retorno. Si la condición cc no se cumple, se incrementa el contador de programa PC, no se realiza el retorno y se continúa con la ejecución de la instrucción siguiente dentro del subprograma.

Ejecución: Condición cumplida: 3 ciclos M; 11 estados T; 3,25 µs 4,00 MHz
Condición no cumplida: 1 ciclo M; 5 estados T; 1,25 µs 4,00 MHz.

Direccionamiento: Indirecto

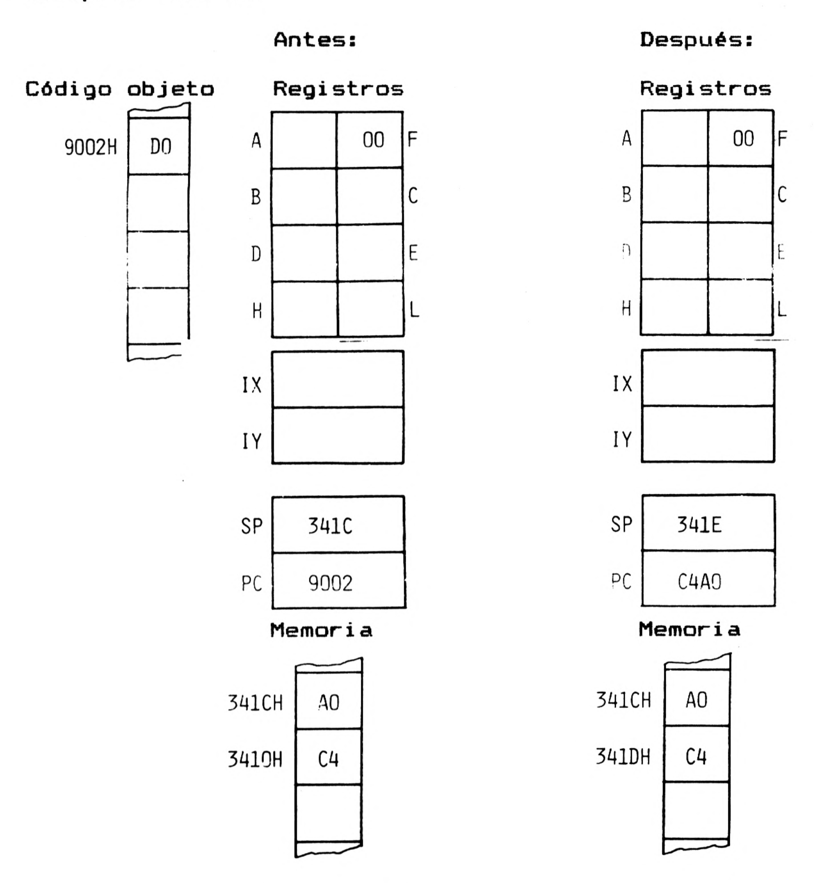
Formato:

1 1-----cc----0 0 0

cc puede ser:

NZ - 000	PO - 100
Z - 001	PE - 101
NC - 010	P - 110
C - 011	M - 111

Ejemplo: RET NC



Flags activados:

Flags activados:

RETI Retorno del programa de tratamientos de interrupt.

Función: <———

Flags: S Z - H - P/V N C

Descripción: El contador del programa PC se carga con las dos últimas entradas de la pila (para más detalles véase la instrucción "POP"). De esta manera podrá continuar el programa con la siguiente instrucción del programa principal. Esta instrucción será reconocida y comprobada por los componentes especiales Zilog. Se tiene de esta manera la posibilidad de obtener interrupts encadenados, ya que los componentes desactivan sus demandas de interrupt al detectar la instrucción de RETI.

Ejecución:

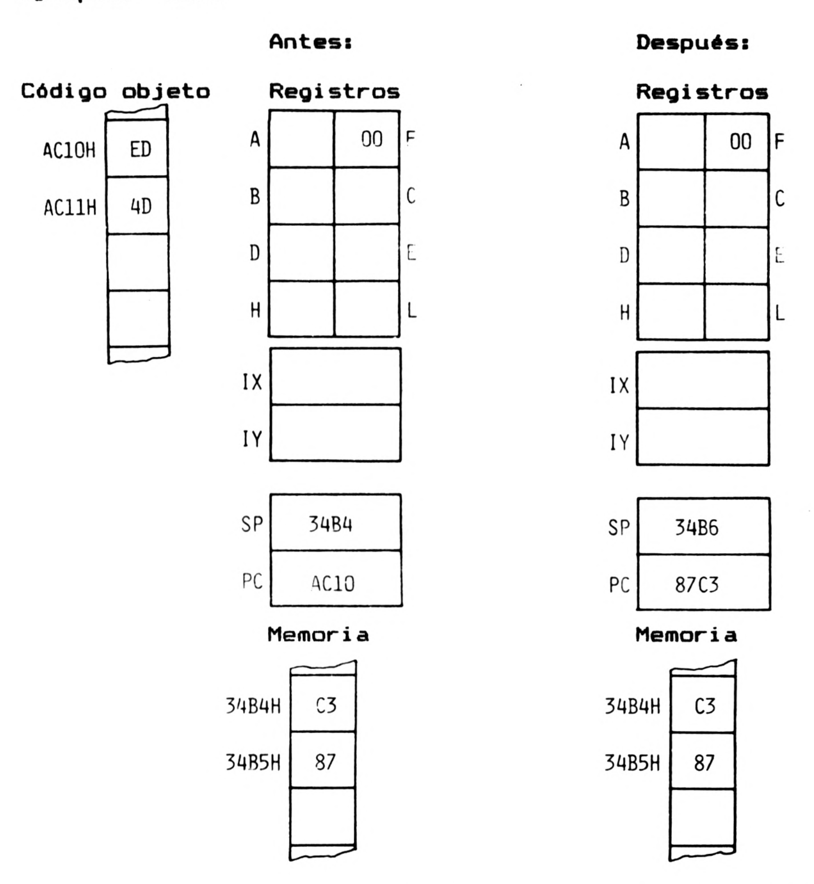
Tiempo de ejecución	:	3,50 µs
Frecuencia de reloj	:	4,00 MHz
Estados de reloj	:	14
Ciclos de máquina	:	4

Direccionamiento: Indirecto

Formato:

Byte 1)	1	1	1	0	1	1	0	1	EDH
Byte 2)	0	1	0	0	1	1	0	1	4DH

Ejemplo: RETI



Flags activados: Flags activados:

RETN — **Retorno de un interrupt sin máscara**

Función: $PC_{inferior}$ -- (SP); $PC_{superior}$ -- (SP+1); SP -- SP + 2; IFF1 -- IFF2

Flags: S Z - H - P/V N C
(ninguna influencia)

Descripción: El contador del programa PC se carga con las dos últimas inserciones de la pila (para más detalles véase la instrucción "POP"). De esta manera el programa continua con la siguiente instrucción del programa principal. Se carga además en primer lugar el contenido del interrupt flip-flop IFF2 después del IFF1 restaurando el estado original del interrupt INT mascarable.

Ejecución:

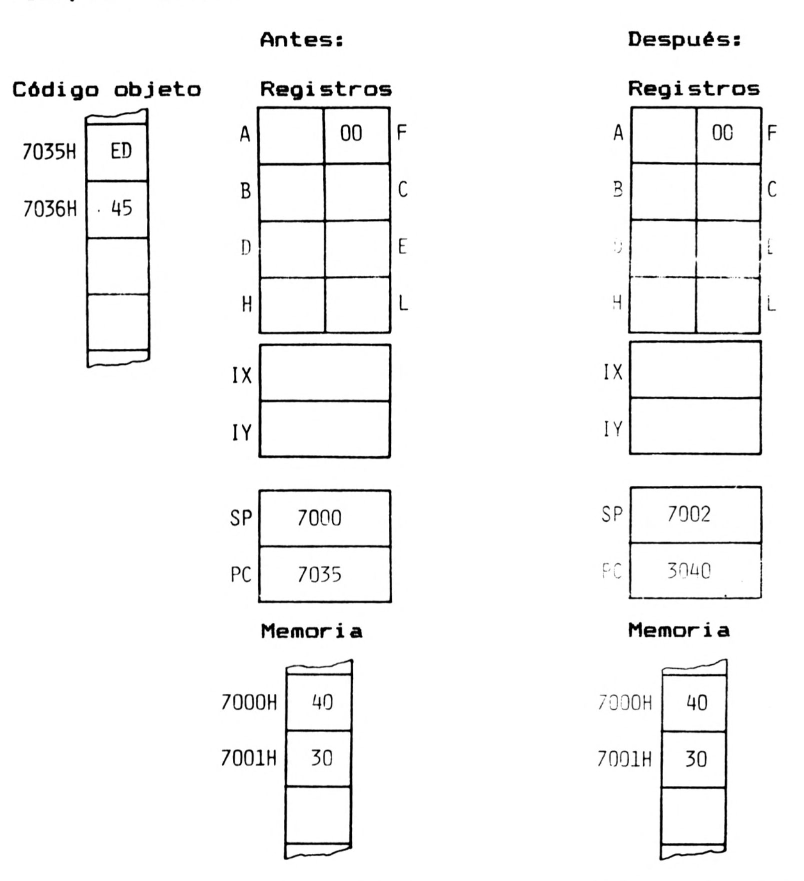
Tiempo de ejecución	:	3,50 µs
Frecuencia de reloj	:	4,00 MHz
Estados de reloj	:	14
Ciclos de máquina	:	4

Direccionamiento: Indirecto

Formato:

Byte 1)	1	1	1	0	1	1	0	1	EDH
Byte 2)	0	1	0	0	0	1	0	1	45H

Ejemplo: RETN



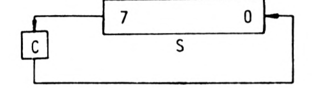
Flags activados:

Flags activados:

RL s

Rotación hacia la izquierda mediante el bit de acarreo (Carry).

Función:



Flags:

S	Z	-	H	-	P/V	N	C
X	X		O		X	O	X

C se activa a través del bit de origen 7.

Descripción: será

El operando activado a través de s desplazado hacia la izquierda. El bit que quedará libre en la posición cero del operando será cargado con el contenido del carryflag, mientras el bit "que se desprenda" de la posición siete se carga en el carryflag. De esta manera se obtendrá una rotación de nueve bits hacia la izquierda.

Ejecución:

s:	µs (4,00 MHz)	Estados T	Ciclos M
r	2,00	8	2
(HL)	3,75	15	4
(IX + d)	5,75	23	6
(IY + d)	5,75	23	6

Direccionamiento: r: implícito/ (HL):indirecto/ (IX d), (IY d): indexado.

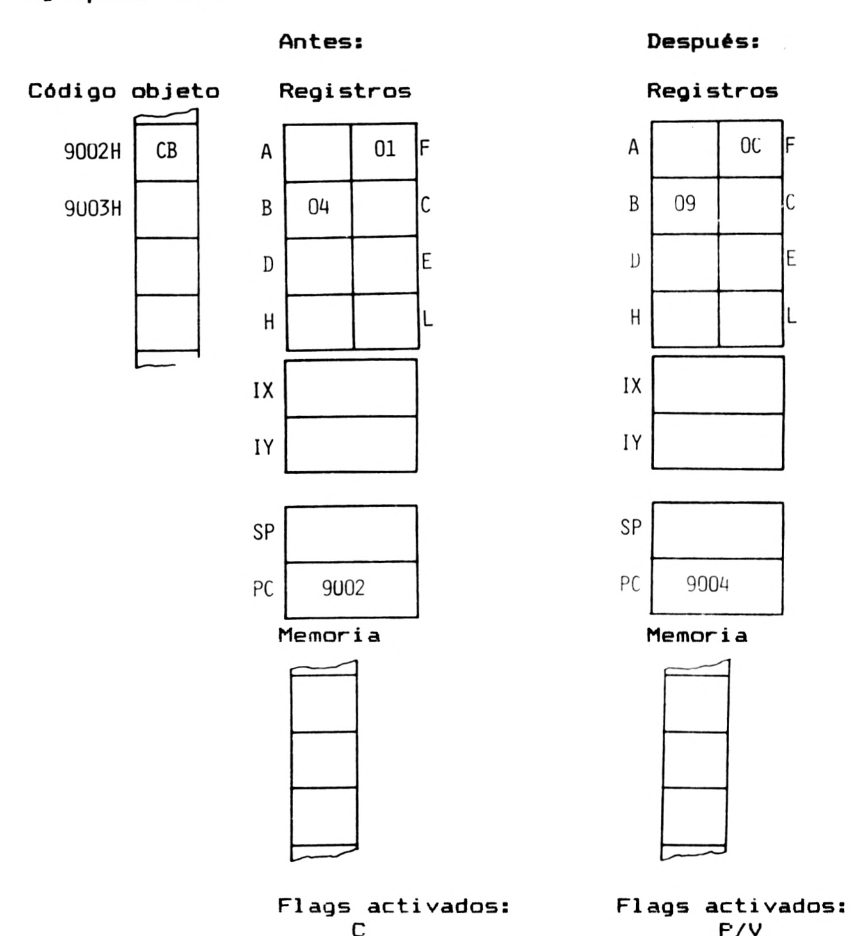
Formato: s:

r:	Byte 1):	1	1	0	0	1	0	1	1	CBH
	Byte 2):	0	0	0	1	0-----r---				
(HL):	Byte 1):	1	1	0	0	1	0	1	1	CBH
	Byte 2):	0	0	0	1	0	1	1	0	16H
(IX + d):	Byte 1):	1	1	0	1	1	1	0	1	DDH
	Byte 2):	1	1	0	0	1	0	1	1	CBH
	Byte 3):	-----------d----------Offset								
	Byte 4):	0	0	0	1	0	1	1	0	16H
(IY + d)	Byte 1):	1	1	1	1	1	1	0	1	FDH
	Byte 2):	1	1	0	0	1	0	1	1	CBH
	Byte 3):	-----------d----------Offset								
	Byte 4):	0	0	0	1	0	1	1	0	16H

r puede ser:

A - 111	E - 011
B - 000	H - 100
C - 001	L - 101
D - 010	

Ejemplo: RL B

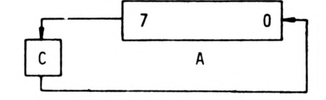


Flags activados: C

Flags activados: P/V

<u>RLA</u> — **Rotación del acumulador hacia la izquierda mediante el bit de acarreo.**

Función:



Flags: S Z - H - P/V N C

0 0 X

C se activa a través del bit 7 de A.

Descripción: El contenido del acumulador A será desplazado hacia la izquierda. El bit liberado en la posición cero del acumulador será cargado con el contenido del carryflag, mientras el bit que se "desprenda" de la posición 7 sea cargado en el carryflag. De este modo se obtendrá una rotación de nueve bits hacia la izquierda.

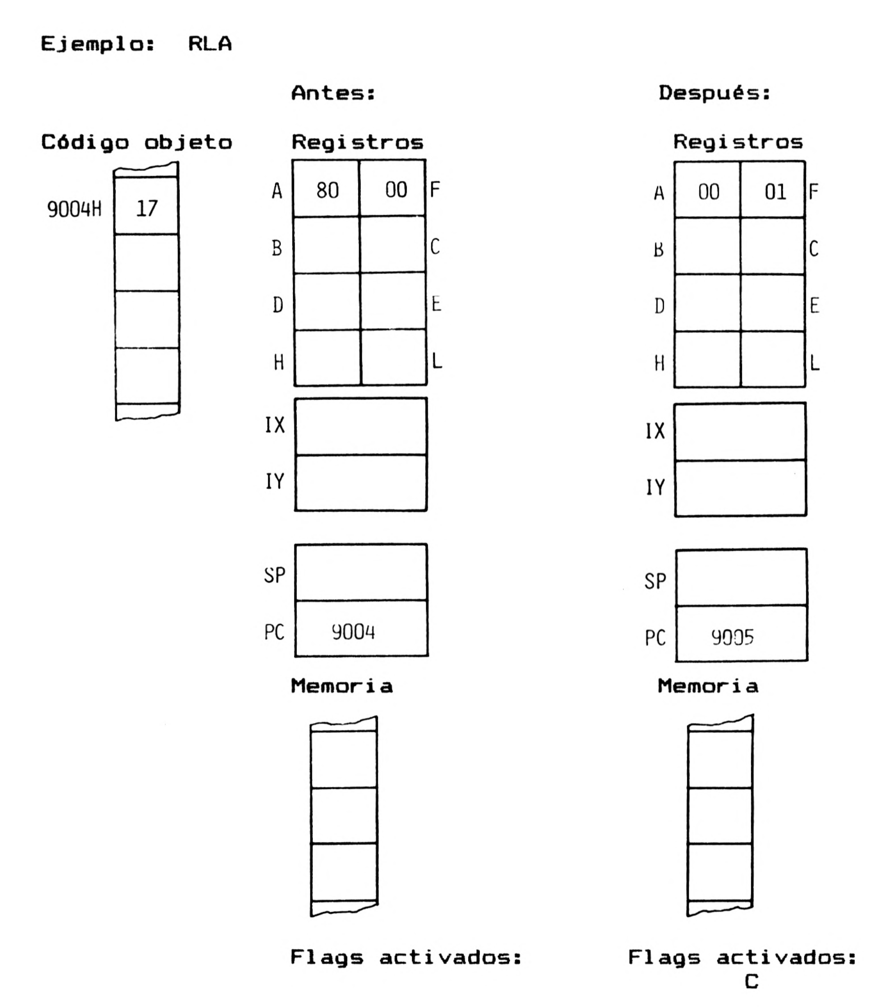
Ejecución:

Tiempo de ejecución	:	1,00 µs
Frecuencia de reloj	:	4,00 MHz
Estados de reloj	:	4
Ciclos de máquina	:	1

Direccionamiento: Implícito

Formato: 0 0 0 1 0 1 1 1 17H

Ejemplo: RLA

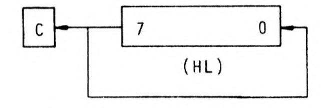


Flags activados:

Flags activados:
C

RLC (HL) — Rotación de la célula de memoria (HL) de forma cíclica hacia la izquierda.

Función:



Flags:

S	Z	-	H	-	P/V	N	C
X	X		0		X	0	X

C se activa a través bit de origen 7.

Descripción: El operando (HL) direccionado indirectamente a través del par de registros HL será desplazado hacia la izquierda, donde el bit que se "desprenda" de la posición siete será cargado en la posición liberada que ocupaba el cero. Al mismo tiempo se cargará el carryflag con el contenido "desprendido" del bit siete. De este modo se obtendrá una rotación de ocho bits hacia la izquierda.

Ejecución:

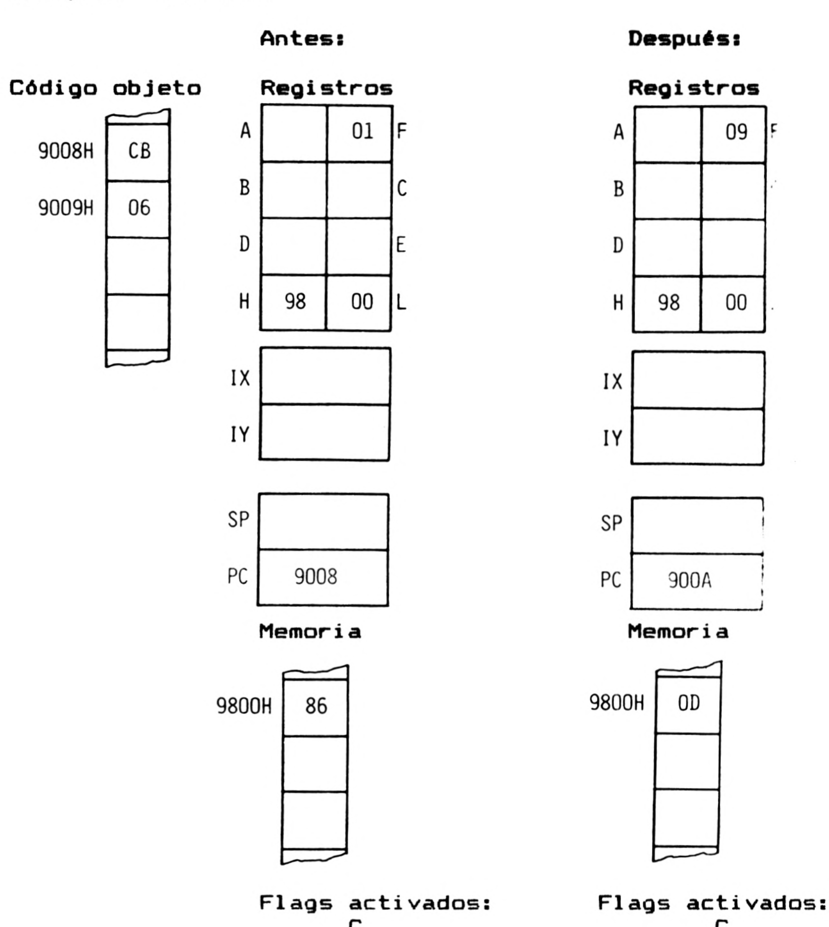
Tiempo de ejecución : 3,75 µs
Frecuencia de reloj : 4,00 MHz
Estados de reloj : 15
Ciclos de máquina : 4

Direccionamiento: Indirecto

Formato:

Byte 1)	1	1	0	0	1	0	1	1	CBH
Byte 2)	0	0	0	0	0	1	1	0	06H

Ejemplo: RLC (HL)



RLC (IX + d) Rotación de la célula de memoria (IX + d) de manera cíclica hacia la izquierda.

Función:

C ← 7 ... 0 ← (IX + d)

Flags:

S	Z	-	H	-	P/V	N	C
X	X		O		X	O	X

C se activa mediante bit 7 original.

Descripción: El operando (IX + d) direccionado indexadamente a través del registro de índices IX y del desplazamiento d, direccional, se desplaza hacia la izquierda, donde el bit que se "desprenda" de la posición siete se carga en la posición liberada ocupada por el cero. Al mismo tiempo se carga también el carryflag con el contenido "desprendido" del bit siete. De este modo se obtiene una rotación de ocho bits hacia la izquierda.

Ejecución:

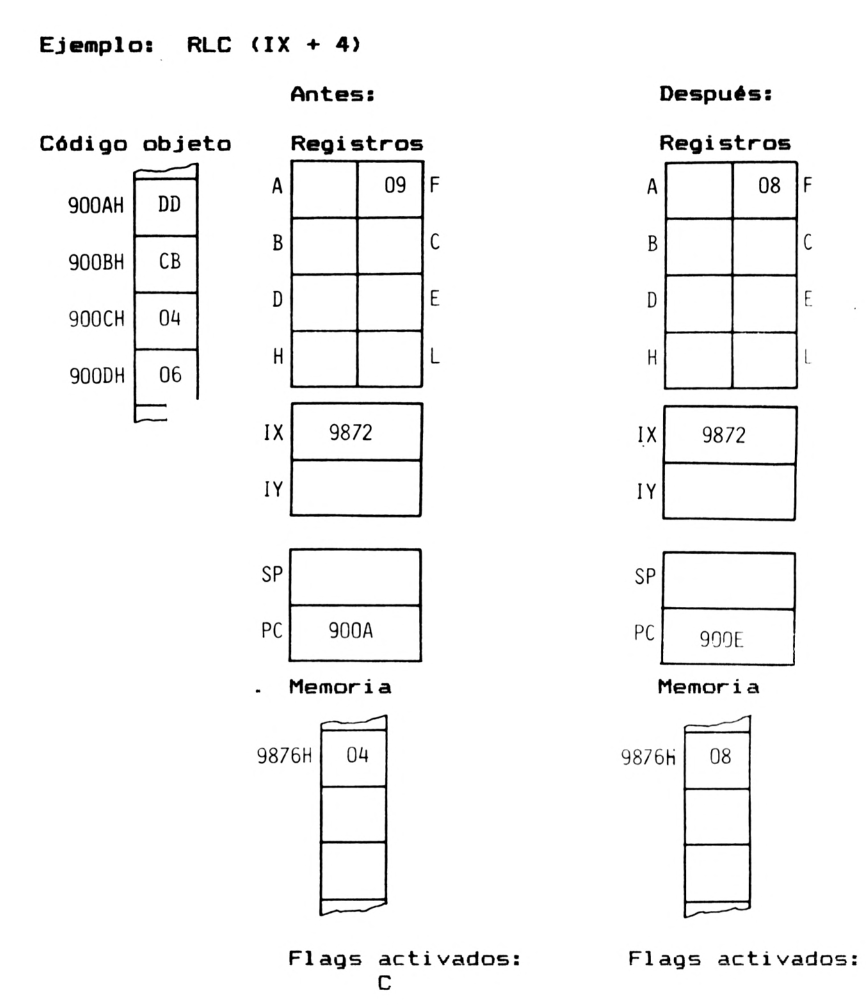
Tiempo de ejecución	:	5,75 µs
Frecuencia de reloj	:	4,00 MHz
Estados de reloj	:	23
Ciclos de máquina	:	6

Direccionamiento: Indexado

Formato:

Byte 1)	1	1	0	1	1	1	0	1	DDH
Byte 2)	1	1	0	0	1	0	1	1	CBH
Byte 3)	-----------d----------Offset								
Byte 4)	0	0	0	0	0	1	1	0	06H

Ejemplo: RLC (IX + 4)



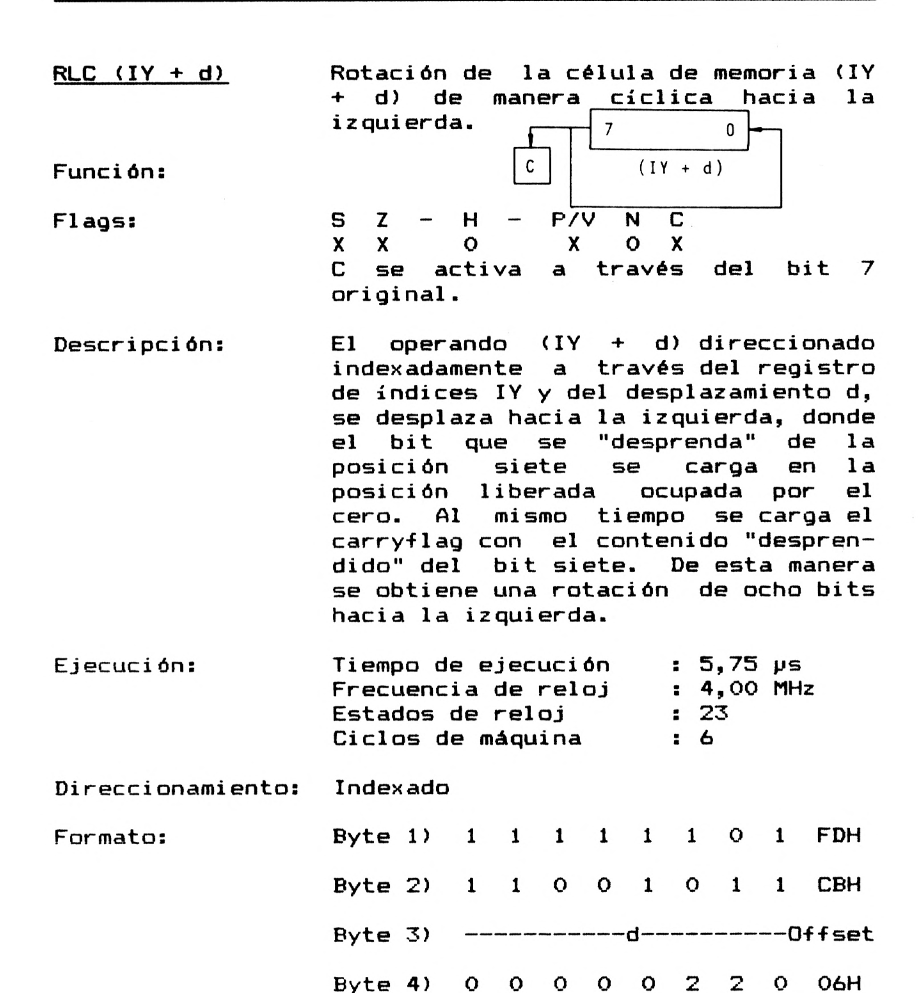
Flags activados:
C

Flags activados:

RLC (IY + d)

Rotación de la célula de memoria (IY + d) de manera cíclica hacia la izquierda.

Función:



Flags:

S	Z	-	H	-	P/V	N	C
X	X		0		X	0	X

C se activa a través del bit 7 original.

Descripción:

El operando (IY + d) direccionado indexadamente a través del registro de índices IY y del desplazamiento d, se desplaza hacia la izquierda, donde el bit que se "desprenda" de la posición siete se carga en la posición liberada ocupada por el cero. Al mismo tiempo se carga el carryflag con el contenido "desprendido" del bit siete. De esta manera se obtiene una rotación de ocho bits hacia la izquierda.

Ejecución:

Tiempo de ejecución : 5,75 µs
Frecuencia de reloj : 4,00 MHz
Estados de reloj : 23
Ciclos de máquina : 6

Direccionamiento: Indexado

Formato:

Byte 1)	1	1	1	1	1	1	0	1	FDH
Byte 2)	1	1	0	0	1	0	1	1	CBH
Byte 3)	-----------			d			----------		Offset
Byte 4)	0	0	0	0	0	2	2	0	06H

Ejemplo: RLC (IY + 4)

Código objeto

Dirección	Contenido
900AH	FD
900BH	CB
900CH	04
900DH	06

Antes:

Registros

A		09	F
B			C
D			E
H			L

IX	
IY	9872

SP	
PC	900A

Memoria

9876H	04

Flags activados:
C

Después:

Registros

A		00	F
B			C
D			E
H			L

IX	
IY	9872

SP	
PC	900E

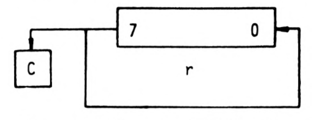
Memoria

9876H	08

Flags activados:

RLC r — **Rotación cíclica hacia la izquierda.**

Función:



Flags:

S	Z	-	H	-	P/V	N	C
X	X		0		X	0	X

C se activa mediante el bit 7 original.

Descripción:

El operando definido a través de r se desplaza hacia la izquierda, donde el bit que se "desprenda" de la posición siete se carga en la posición liberada ocupada por cero. Al mismo tiempo se carga el carry-flag con el contenido "desprendido" del bit siete. De este modo se obtendrá una rotación de ocho bits hacia la izquierda.

Ejecución:

Tiempo de ejecución	:	2,00 µs
Frecuencia de reloj	:	4,00 MHz
Estados de reloj	:	8
Ciclos de máquina	:	2

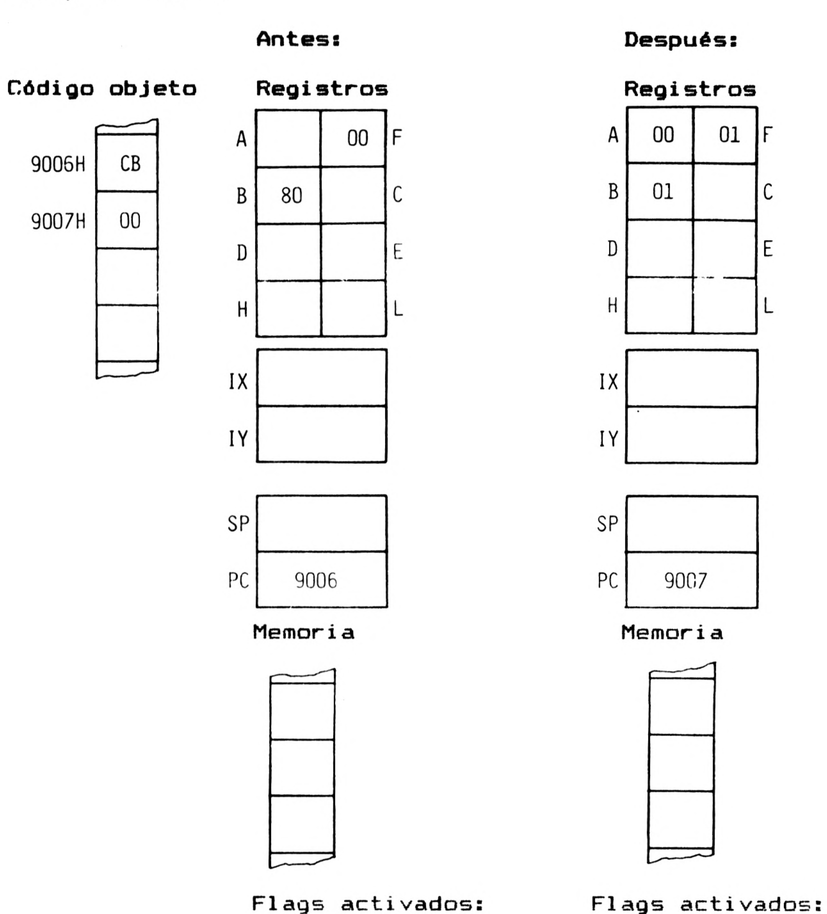
Direccionamiento: Implícito

Formato:	Byte 1)	1	1	0	0	1	0	1	1	CBH
	Byte 2)	0	0	0	0	0-----r---				

r puede ser:

B - 000	H - 100
C - 001	L - 101
D - 010	

Ejemplo: RLC B

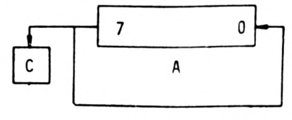


Flags activados:

Flags activados:
C

RLCA — Rotación cíclica del acumulador hacia la izquierda.

Función:



Flags:	S	Z	-	H	-	P/V	N	C
				0			0	X

C activa el bit 7 de A.

Descripción: El contenido del acumulador será desplazado hacia la izquierda, en donde el bit que se "desprenda" de la posición siete será cargado en la posición liberada de la posición cero. Al mismo tiempo se cargará el carryflag con el contenido "desprendido" del bit siete. De esta manera se obtendrá una rotación de ocho bits hacia la izquierda.

Ejecución:

Tiempo de ejecución	:	1,00 µs
Frecuencia de reloj	:	4,00 MHz
Estados de reloj	:	4
Ciclos de máquina	:	1

Direccionamiento: Implícito

Formato: 0 0 0 0 0 1 1 1 07H

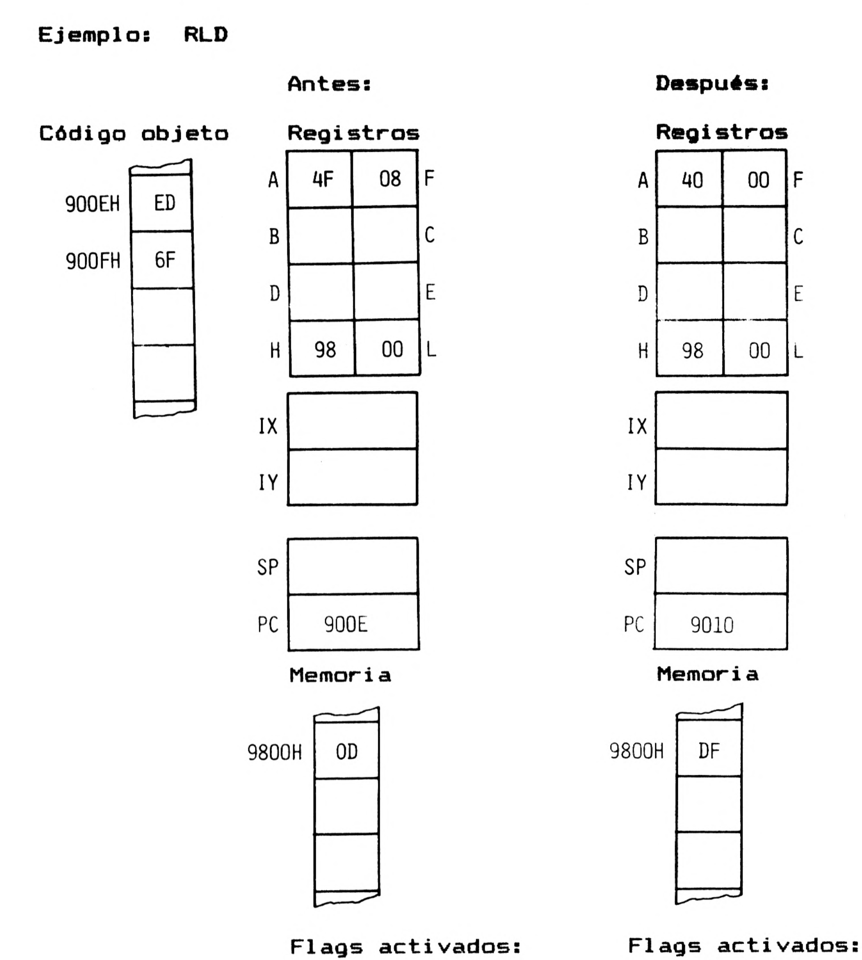
Ejemplo: RLCA

	Antes:	Después:
Código objeto	**Registros**	**Registros**
9005H: 07	A 00, F 01	A 00, F 00
	B, C	B, C
	D, E	D, E
	H, L	H, L
	IX	IX
	IY	IY
	SP	SP
	PC 9005	PC 9006
	Memoria	**Memoria**
	Flags activados: C	Flags activados:

RLD — **Rotación decimal hacia la izquierda.**

Función:	A 7 4 3 0 7 4 3 0 HL
Flags:	S Z - H - P/V - N C
	X X O X O
Descripción:	El nibble inferior de la posición de memoria direccionada indirectamente a través del par de registros HL será desplazado a su nibble superior, al mismo tiempo se desplaza el nibble superior al nibble inferior del acumulador y a su vez el nibble inferior del acumulador se desplaza en el nibble inferior de la posición de memoria (HL). De esta manera no se "pierde" ningún nibble y el efecto corresponde a una rotación decimal de tres posiciones hacia la izquierda.
Ejecución:	Tiempo de ejecución : 4,50 µs Frecuencia de reloj : 4,00 MHz Estados de reloj : 18 Ciclos de máquina : 5
Direccionamiento:	Indirecto
Formato:	Byte 1) 1 1 1 0 1 1 0 1 EDH Byte 2) 0 1 1 0 1 1 1 1 6FH

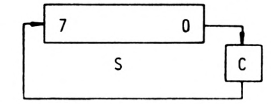
Ejemplo: RLD



Flags activados: Flags activados:

RR s — **Rotación hacia la derecha mediante el bit C de acarreo.**

Función:



Flags:

S	Z	-	H	-	P/V	N	C
X	X		O		X	O	X

C se activa mediante el bit O original O.

Descripción:

El operando definido a través de s se desplaza hacia la derecha. El bit liberado del operando de la posición siete se carga con el contenido del carryflag, mientras el bit que se "desprenda" de la posición cero se cargue en el carryflag. De esta manera se obtendrá una rotación de nueve bits hacia la derecha.

Ejecución:

s	µs (4,00 MHz)	Estados T	Ciclos M
r	2,00	8	2
(HL)	3,75	15	4
(IX+d)	5,75	23	6
(IY+d)	5,75	23	6

Direccionamiento: r: implícito/ (HL): indirecto/(IX+d), (IY+d): indexado.

Formato:											
r:	Byte 1):	1	1	0	0	1	0	1	1	CBH
	Byte 2):	0	0	0	1	1	----r---			
(HL):	Byte 1):	1	1	0	0	1	0	1	1	CBH
	Byte 2):	0	0	0	1	1	1	1	0	1EH
(IX + d):	Byte 1):	1	1	0	1	1	1	0	1	DDH
	Byte 2):	1	1	0	0	1	0	1	1	CBH
	Byte 3):	----------d-----------Offset								
	Byte 4):	0	0	0	1	1	1	1	0	1EH
(IY + d):	Byte 1):	1	1	1	1	1	1	0	1	FDH
	Byte 2):	1	1	0	0	1	0	1	1	CBH
	Byte 3):	----------d-----------Offset								
	Byte 4):	0	0	0	1	1	1	1	0	1EH

r puede ser :

A - 111

B - 000

C - 001

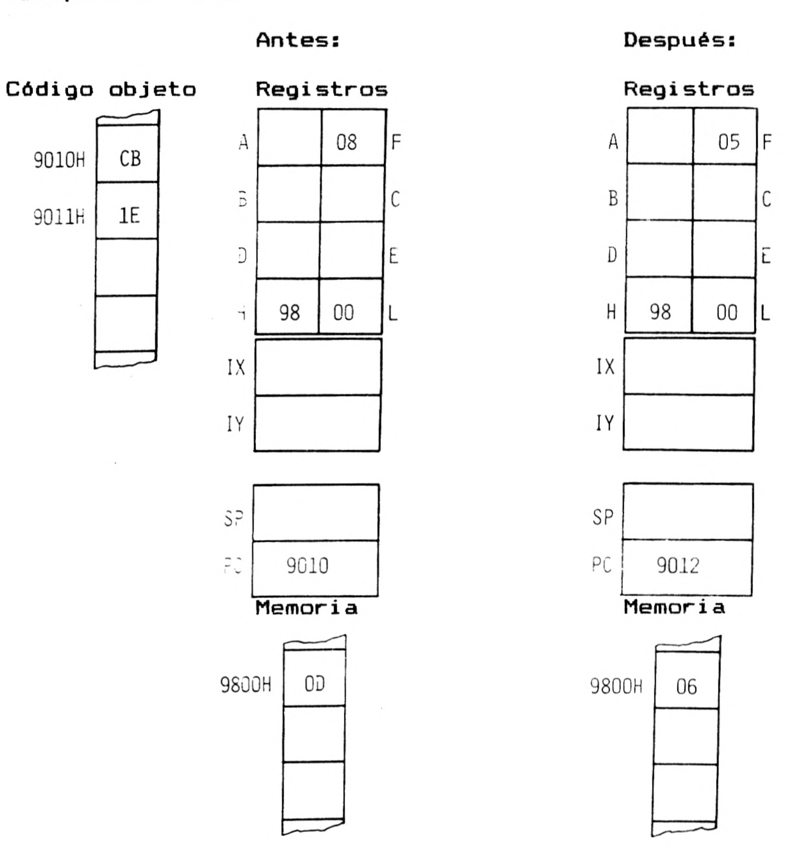
D - 010

E - 011

H - 100

L - 101

Ejemplo: RR (HL)



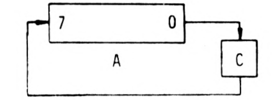
Flags activados:

Flags activados:

P/V

RRA — Rotación del acumulador hacia la derecha mediante el bit de acarreo.

Función:



Flags:

S	Z	-	H	-	P/V	N	C
			0			0	X

C se activa a través del bit 0 de A.

Descripción: El contenido del acumulador A se desplaza hacia la derecha. El bit liberado del acumulador en posición siete se carga con el contenido del carryflag, mientras el bit que se "desprenda" de la posición cero se carga en el carryflag. De esta manera se obtendrá una rotación de nueve bits hacia la derecha.

Ejecución:

Tiempo de ejecución : 4,00 µs
Frecuencia de reloj : 4,00 MHz
Estados de reloj : 4
Ciclos de máquina : 1

Direccionamiento: Implícito

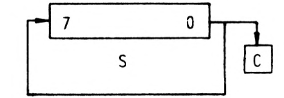
Formato: 0 0 0 1 1 1 1 1 1FH

Ejemplo: RRA

	Antes:	Después:
Código objeto	Registros	Registros
9012H: 1F	A 80, F 00	A 40, F 00
	B, C	B, C
	D, E	D, E
	H, L	H, L
	IX	IX
	IY	IY
	SP	SP
	PC 9012	PC 9013
	Memoria	Memoria
	Flags activados:	Flags activados:

RRC s Rotación hacia la derecha.

Función:



Flags:

S	Z	-	H	-	P/V	N	C
X	X		O		X	O	X

La C será activa a través Bit 0 del origen.

Descripción: El contenido de la posición determinada por el operando efectuará una rotación hacia la derecha, almacenando el resultado en la posición primitiva. El contenido del bit cero se desplaza al flag de acarreo y al mismo tiempo al bit siete. La s queda definida en la descripción de la instrucción RLC.

Ejecución:

s	µs (4,00 Mhz)	Estados T	Ciclos M
r	2,00	8	2
(HL)	3,75	15	4
(IX+d)	5,75	23	6
(IY+d)	5,75	23	6

Direccionamiento: r: implícito/ (HL): indirecto/(IX+d); (IY+d): indexado.

Formato:

s: s puede ser r,(HL), (IX+d),(IY+d)

r:	Byte 1)	1	1	0	0	1	0	1	1	CBH
	Byte 2)	0	0	0	0	1----r----				
(HL):	Byte 1)	1	1	0	0	1	0	1	1	CBH
	Byte 2)	0	0	0	0	1	1	1	0	0EH
(IX+d):	Byte 1)	1	1	0	1	1	1	0	1	DDH
	Byte 2)	1	1	0	0	1	0	1	1	CBH
	Byte 3)	----------d-----------								Offset
	Byte 4)	0	0	0	0	1	1	1	0	0EH
(IY+d):	Byte 1)	1	1	1	1	1	1	0	1	FDH
	Byte 2)	1	1	0	0	1	0	1	1	CBH
	Byte 3)	----------d-----------								Offset
	Byte 4)	0	0	0	0	1	1	1	0	0EH

r puede ser :

A - 111	E - 011
B - 000	H - 100
C - 011	L - 101
D - 010	

Ejemplo: RRC (HL)

Antes:

Código objeto

Dirección	Contenido
9013H	CB
9014H	0E

Registros

A		00	F
B			C
D			E
H	98	00	L
IX			
IY			
SP			
PC	9013		

Memoria

Dirección	Contenido
9800H	07

Flags activados:

Después:

Registros

A		81	F
B			C
D			E
H			L
IX			
IY			
SP			
PC	9015		

Memoria

Dirección	Contenido
9800H	83

Flags activados:

S, C

RRCA — **Rotación del acumulador hacia la derecha.**

Función:

7 0
A
C

Flags:

S	Z	-	H	-	P/V	N	C
			O			O	X

C será activa a través del bit 0 de A.

Descripción: El contenido del acumulador se desplaza hacia la derecha, en donde el bit que se ha "desprendido" de la posición cero se carga en la posición siete liberada. Al mismo tiempo se carga el carryflag con el contenido "desprendido" del bit cero. De esta manera se obtendrá una rotación de ocho bits hacia la derecha.

Ejecución:

Tiempo de ejecución	: 1,00 µs
Frecuencia de reloj	: 4,00 MHz
Estados de reloj	: 4
Ciclos de máquina	: 1

Direccionamiento: Implícito

Formato: 0 0 0 0 1 1 1 1 OFH

Ejemplo: RRCA

Código objeto

Dirección	Contenido
9015H	0F

Antes:

Registros

A	40	81	F
B			C
D			E
H			L
IX			
IY			
SP			
PC	9015		

Memoria

Flags activados:
S,C

Después:

Registros

A	20	A0	F
B			C
D			E
H			L
IX			
IY			
SP			
PC	9016		

Memoria

Flags activados:
S

RRD — **Rotación decimal hacia la derecha.**

Función:

A | 7 4 | 3 0 | | 7 4 | 3 0 | HL

Flags:

S	Z	-	H	-	P/V	N	C
X	X		0		X	0	

Descripción: El nibble superior de la posición de memoria direccionada indirectamente a través del par de registros HL se desplaza a su nibble inferior, mientras al mismo tiempo el nibble inferior se desplaza al nibble inferior del acumulador y a su vez el nibble inferior del acumulador se desplaza al nibble superior de la posición de memoria (HL). De esta manera no se "perderá" ningún nibble y el efecto corresponderá a una rotación decimal de 3 posiciones hacia la derecha.

Ejecución:

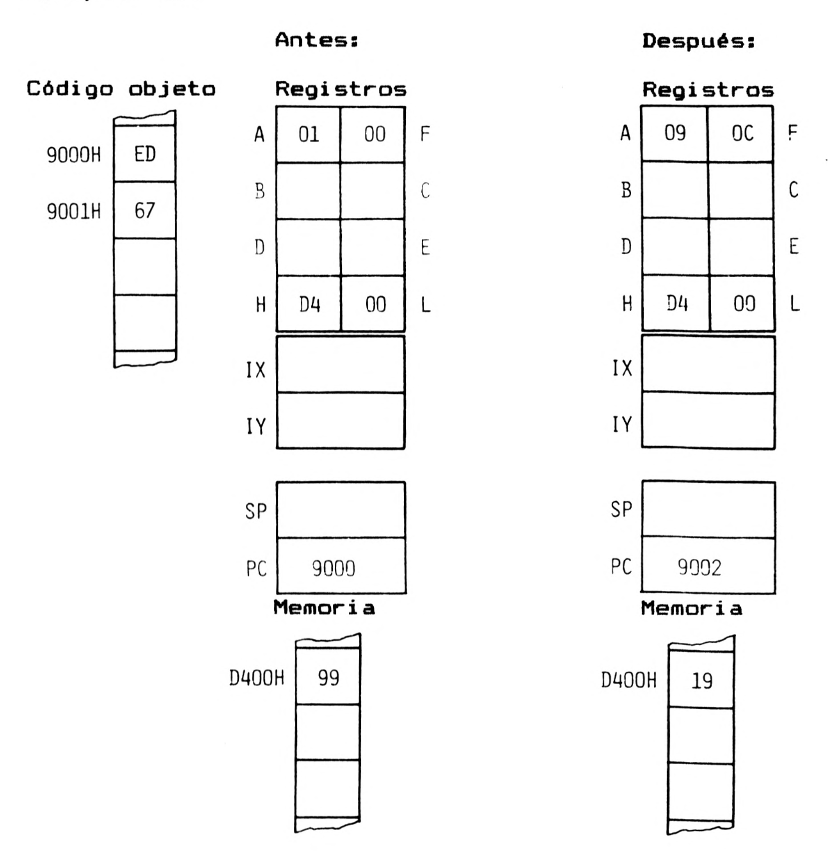
Tiempo de ejecución : 4,50 µs
Frecuencia de reloj : 4,00 MHz
Estados de reloj : 18
Ciclos de máquina : 5

Direccionamiento: Indirecto

Formato:

Byte 1)	1	1	1	0	1	1	0	1	EDH
Byte 2)	0	1	1	0	0	1	1	1	67H

Ejemplo: RRD



Flags activados:

Flags activados:
P/V

RST p — Restart en p.

Función: $(SP-1) \leftarrow PC_{superior}$; $(SP-2) \leftarrow PC_{inferior}$; $SP \leftarrow SP - 2$; $PC_{superior} \leftarrow 0$; $PC_{inferior} \leftarrow p$

Flags: S Z - H - P/V N C
(ninguna influencia)

Descripción: El programa bifurca a la dirección p del registro (Zero Page). En primer lugar se almacena el contenido del contador del programa en la pila (véase más detalles en la instrucción PUSH). El comando puede enviarse al bus de datos depués de una interrupción a través del componente que provocó la interrupción, por ejemplo en modalidad 0 de interrupt. Otra posibilidad de aplicación sería utilizar esta instrucción como interrupt de software, aplicando la instrucción en una posición adecuada del programa.

Ejecución:

Tiempo de ejecución	: 2,75 µs
Frecuencia de reloj	: 4,00 MHz
Estados de reloj	: 11
Ciclos de máquina	: 3

Direccionamiento: Indirecto

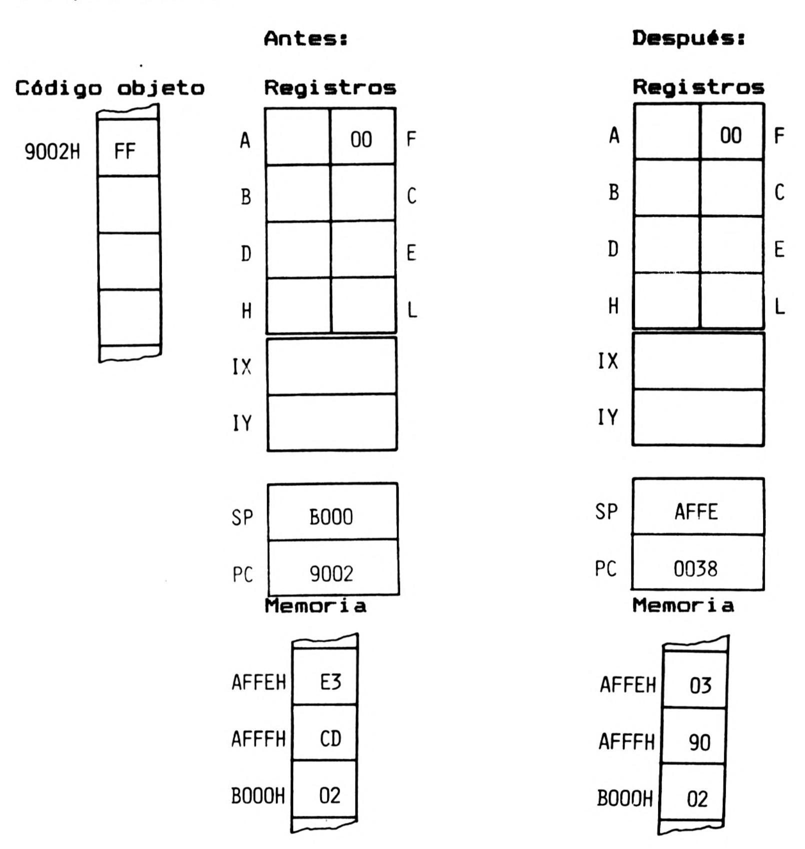
Formato:

1 1-----p-----1 1 1

p puede ser :

00H - 000	20H - 100
08H - 001	28H - 101
10H - 010	30H - 110
18H - 011	38H - 111

Ejemplo: RST 38



Flags activados: Flags activados:

SBC A,s — Resta del acumulador el operando s indicado con acarreo.

Función: A <—— A - s - C

Flags:

S	Z	-	H	-	P/V	N	C
X	X		X		X	1	X

Descripción: Del contenido del acumulador A se sustraen el operando s y el contenido del carryflag C. El resultado se almacena en el acumulador A.

Ejecución:

s:	µs (4,00 Mhz)	Estados T	Ciclos M
r	1,00	4	1
n	1,75	7	2
(HL)	1,75	7	2
(IX+d)	4,75	19	5
(IY+d)	4,75	19	5

Direccionamiento: r: implícito/ n: inmediato/ (HL): indirecto/ (IX+d),(IY+d): indexado

Formato:		s: puede ser r,n,(HL),(IX+d), or (IY+d)									
r:			1	0	0	1	1-----r---				
n:	Byte 1):		1	1	0	1	1	1	1	0	DEH
	Byte 2):		---------n------------Datos inmed.								
(HL):	Byte 1):		1	0	0	1	1	1	1	0	9EH
(IX+d):	Byte 1):		1	1	0	1	1	1	0	1	DDH
	Byte 2):		1	0	0	1	1	1	1	0	9EH
	Byte 3):		---------d------------Offset								
(IY+d):	Byte 1):		1	1	1	1	1	1	0	1	FDH
	Byte 2):		1	0	0	1	1	1	1	0	9EH
	Byte 3):		---------d------------Offset								

r puede ser:

A - 111	E - 011
B - 000	H - 100
C - 001	L - 101
D - 010	

Ejemplo: SBC A,(IX + 2)

Antes:

Código objeto

9000H	FD
9001H	9E
9002H	02

Registros

A	09	00	F
B			C
D			E
H			L
IX	803C		
IY			
SP			
PC	9000		

Memoria

803EH	04

Flags activados:

Después:

Registros

A	05	02	F
B			C
D			E
H			L
IX	803C		
IY			
SP			
PC	9003		

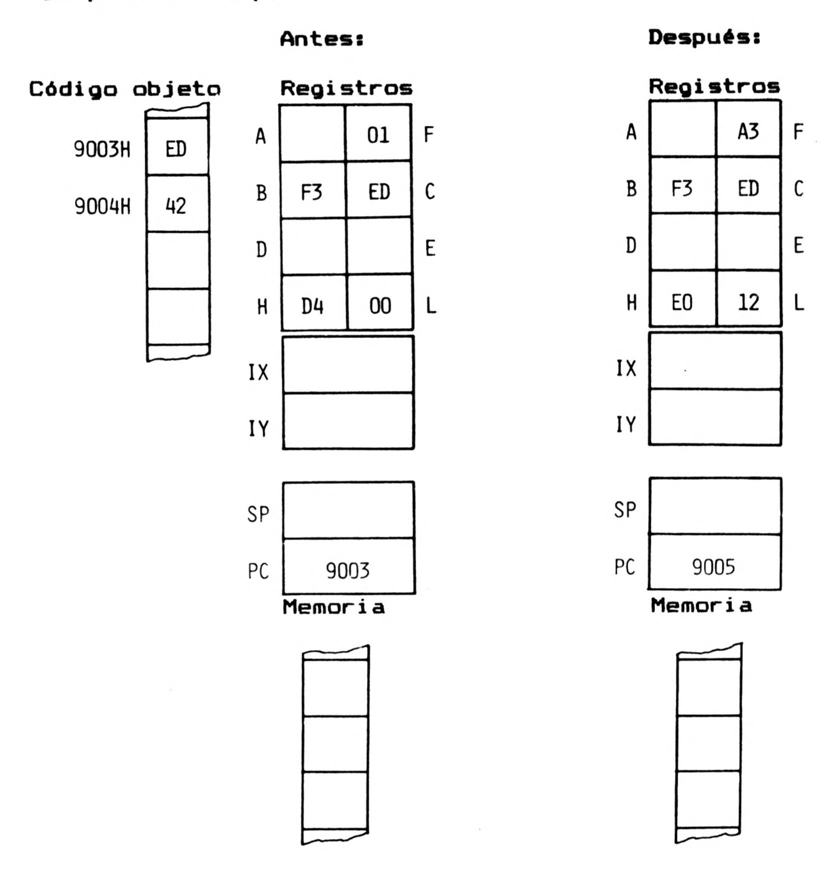
Memoria

803EH	04

Flags activados:
N

<u>SBC HL,ss</u>	Sustrae del par de registros HL el par de registros indicado con el acarreo.
Función:	HL <— HL - ss - C
Flags:	S Z - H - P/V N C X X X X 1 X H será activo si hay acarreo del bit 12. C será activo si hay acarreo.
Descripción:	Del contenido del par de registros HL se restarán el operando ss de 16 bits y el contenido del carryflag C. El resultado será almacenado en el par de registros HL.
Ejecución:	Tiempo de ejecución : 3,75 µs Frecuencia de reloj : 4,00 MHz Estados de reloj : 15 Ciclos de máquina : 4
Direccionamiento:	Implícito
Formato:	Byte 1) 1 1 1 0 1 1 0 1 EDH Byte 2) 0 1 S S 0 0 1 0 ss puede ser: BC - 00 HL - 10 DE - 01 SP - 11

Ejemplo: SBC HL,BC



Flags activados:
C

Flags activados:
S,N,C

SCF Activa el flag de acarreo.

Función: C <—— 1

Flags:

S	Z	-	H	-	P/V	N	C
			0			0	1

Descripción: El carryflag del registro de estado se activa en lógico 1.

Ejecución:

Tiempo de ejecución : 1,00 µs
Frecuencia de reloj : 4,00 MHz
Estados de reloj : 4
Ciclos de máquina : 1

Direccionamiento: Implícito

Formato: 0 0 1 1 0 1 1 1 37H

SET b,s Activa el bit b del operando s.

Función: s_b <— 1

Flags: S Z - H - P/V N C
(ninguna influencia)

Descripción: El bit especificado a través de b del operando determinado mediante s será activado (a lógico 1).

Ejecución:

s	µs (4,00 Mhz)	Estados T	Ciclos M
r	2,00	8	2
(HL)	3,75	15	4
(IX+d)	5,75	23	6
(IY+d)	5,75	23	6

Direccionamiento: r: implícito/ (HL): indirecto/(IX+d), (IY+d): indexado.

Formato: s:

r:	Byte 1):	1	1	0	0	1	0	1	1	CBH
	Byte 2):	1	1-----b--------r---							
(HL):	Byte 1):	1	1	0	0	1	0	1	1	CBH
	Byte 2):	1	1-----b-----1					1	0	
(IX+d):	Byte 1):	1	1	0	1	1	1	0	1	DDH
	Byte 2):	1	1	0	0	1	0	1	1	CBH
	Byte 3):	-----------d----------Offset								
	Byte 4):	1	1-----b-----1					1	0	
(IY+d):	Byte 1):	1	1	1	1	1	1	0	1	FDH
	Byte 2):	1	1	0	0	1	0	1	1	CBH
	Byte 3):	-----------d----------Offset								
	Byte 4):	1	1-----b-----1					1	0	

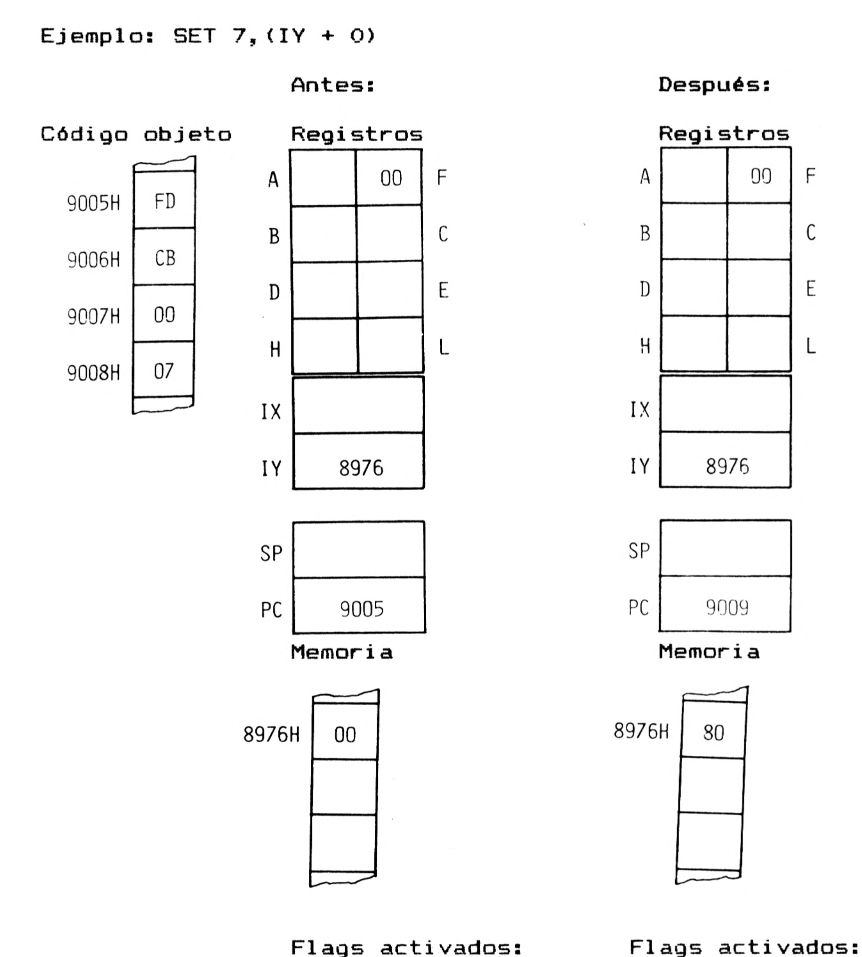
r puede ser:

A - 111	E - 001
B - 000	H - 100
C - 001	L - 101
D - 010	

b puede ser:

0 - 000	4 - 100
1 - 001	5 - 101
2 - 010	6 - 110
3 - 011	7 - 111

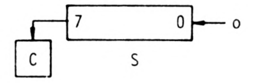
Ejemplo: SET 7,(IY + 0)



Flags activados: Flags activados:

SLA s — Desplazamiento aritmético hacia la izquierda del operando s.

Función:



Flags:

S	Z	-	H	-	P/V	N	C
X	X		O		X	O	X

C será activa con el bit 7 original.

Descripción: El operando definido a través de s se desplaza hacia la izquierda, donde el bit que se ha "desprendido" de la posición siete se carga en el carryflag. La posición cero liberada se rellena con lógico cero.

Ejecución:

s	µs (4,00 Mhz)	Estados T	Ciclos M
r	2,00	8	2
(HL)	3,75	15	4
(IX+d)	5,75	23	6
(IY+d)	5,75	23	6

Direccionamiento: r: implícito/ (HL): indirecto/(IX+d), (IY+d): indexado.

Formato:

s:

r:	Byte 1):	1	1	0	0	1	0	1	1	CBH
	Byte 2):	0	0	1	0	0-----r---				
(HL):	Byte 1):	1	1	0	0	1	0	1	1	CBH
	Byte 2):	0	0	1	0	0	1	1	0	26H
(IX + d):	Byte 1):	1	1	0	1	1	1	0	1	DDH
	Byte 2):	1	1	0	0	1	0	1	1	CBH
	Byte 3):	-----------d-----------Offset								
	Byte 4):	0	0	1	0	0	1	1	0	26H
(IY + d):	Byte 1):	1	1	1	1	1	1	0	1	FDH
	Byte 2):	1	1	0	0	1	0	1	1	CBH
	Byte 3):	-----------d-----------Offset								
	Byte 4):	0	0	1	0	0	1	1	0	26H

r puede ser:

A - 111

B - 000

C - 001

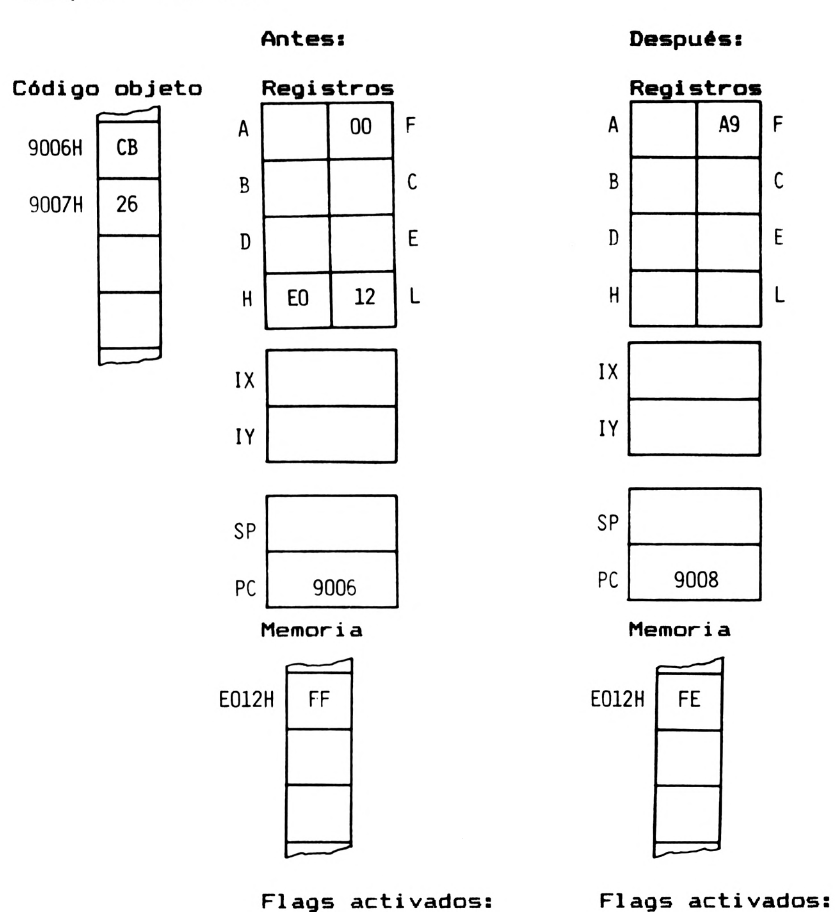
D - 010

E - 011

H - 100

L - 101

Ejemplo: SLA (HL)

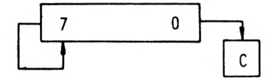


Flags activados:

Flags activados:
S,C

SRA s — Desplazamiento aritmético hacia la derecha del operando s.

Función:



Flags:

S	Z	-	H	-	P/V	N	C
X	X		0		X	0	X

C se activa con el bit 0 del original.

Descripción: El operando definido a través de s se desplaza hacia la derecha, donde el bit que se "desprende" de la posición cero se carga en el carry-flag. El estado de la posición 7 permanecerá invariable.

Ejecución:

s	µs(4,00 MHz)	Estados T	Ciclos M
r	2,00	8	2
(HL)	3,75	15	4
(IX + d)	5,75	23	6
(IY + d)	5,75	23	6

Direccionamiento: r: implícito/ (HL): indirecto/(IX+d), (IY+d): indexado.

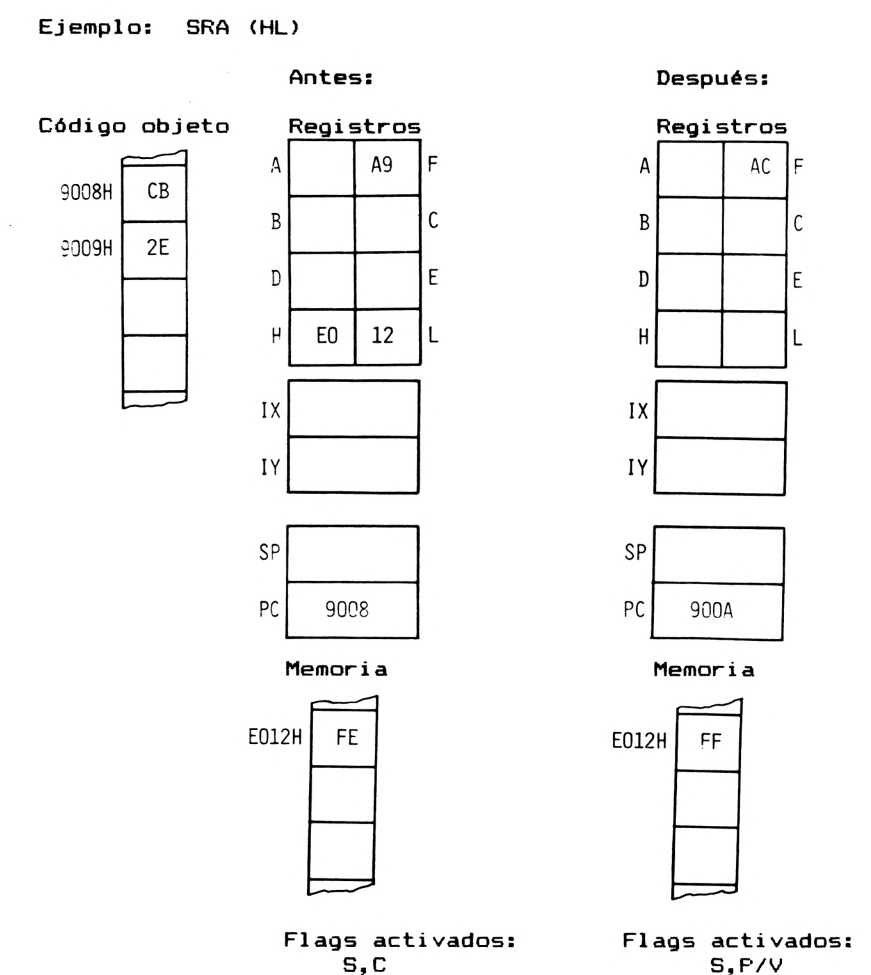
Formato: s:

r:	Byte 1):	1	1	0	0	1	0	1	1	CBH
	Byte 2):	0	0	1	0	1-----r---				
(HL):	Byte 1):	1	1	0	0	1	0	1	1	CBH
	Byte 2):	0	0	1	0	1	1	1	0	2EH
(IX + d):	Byte 1):	1	1	0	1	1	1	0	1	DDH
	Byte 2):	1	1	0	0	1	0	1	1	CBH
	Byte 3):	-----------d----------Offset								
	Byte 4):	0	0	1	0	1	1	1	0	2EH
(IY + d):	Byte 1):	1	1	1	1	1	1	0	1	FDH
	Byte 2):	1	1	0	0	1	0	1	1	CBH
	Byte 3):	-----------d----------Offset								
	Byte 4):	0	0	1	0	1	1	1	0	2EH

r puede ser:

A - 111	E - 011
B - 000	H - 100
C - 001	L - 101
D - 010	

Ejemplo: SRA (HL)



Flags activados:
S,C

Flags activados:
S,P/V

Formato: s:

r:	Byte 1):	1	1	0	0	1	0	1	1	CBH
	Byte 2):	0	0	1	0	0-----r---				
(HL):	Byte 1):	1	1	0	0	1	0	1	1	CBH
	Byte 2):	0	0	1	1	1	1	1	0	36H
(IX + d):	Byte 1):	1	1	0	1	1	1	0	1	DDH
	Byte 2):	1	1	0	0	1	0	1	1	CBH
	Byte 3):	----------d-----------Offset								
	Byte 4):	0	0	1	1	1	1	1	0	3EH
(IY + d):	Byte 1):	1	1	1	1	1	1	0	1	FDH
	Byte 2):	1	1	0	0	1	0	1	1	CBH
	Byte 3):	----------d-----------Offset								
	Byte 4):	0	0	1	1	1	1	1	0	3EH

r puede ser:

A - 111

B - 000

C - 001

D - 010

E - 011

H - 100

L - 101

SRL s — **Desplazamiento lógico hacia la derecha del operando s.**

Función: 0 7 0

s c

Flags: S Z - H - P/V N C

X X 0 X 0 X

C se activa con el bit 7 original.

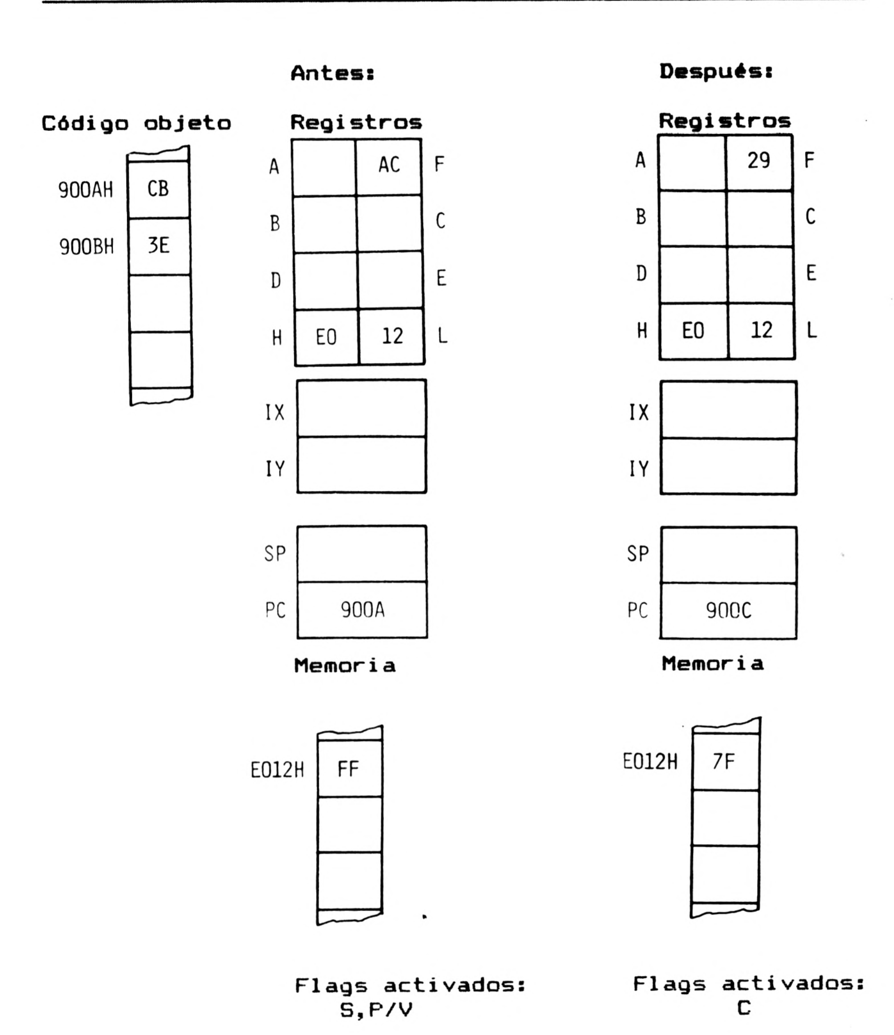
Descripción: El operando determinado a través de s se desplaza hacia la derecha, donde el bit que se "desprende" de la posición cero se carga en el carry-flag. La posición 7 liberada se rellena con cero lógico.

Ejecución:

s	μs (4,00 MHz)	Estados T	Ciclos M
r	2,00	8	2
(HL)	3,75	15	4
(IX + d)	5,75	23	6
(IY + d)	5,75	23	6

Direccionamiento: r: implícito/ (HL): indirecto/(IX+d), (IY+d): indexado.

Ejemplo: SRL (HL)



Flags activados:
S,P/V

Flags activados:
C

SUB s Sustrae del acumulador el operando s indicado.

Función: A <— A - s

Flags:

S	Z	-	H	-	P/V	N	C
X	X		X		X	1	X

Descripción: Del contenido del acumulador A se resta el operando determinado a través de s.

Ejecución:

s	µs(4,00 Mhz)	Estados T	Ciclos M
r	1,00	4	1
n	1,75	7	2
(HL)	1,75	7	2
(IX+d)	4,75	19	5
(IY+d)	4,75	19	5

Direccionamiento: r: implícito/ n: inmediato/ (HL): indirecto/ (IXX+d),(IY+d): indexado.

Formato: s: puede ser r,n,(HL),(IX+d) ó (IY+d)

r:		1	0	0	1	0-----r---				
n:	Byte 1):	1	1	0	1	0	1	1	0	D6H
	Byte 2):	-----------n-----------Datos inmed.								
(HL):		1	0	0	1	0	1	1	0	96H
(IX + d):	Byte 1):	1	1	0	1	1	1	0	1	DDH
	Byte 2):	1	0	0	1	0	1	1	0	96H
	Byte 3):	-----------d-----------Offset								
(IY + d):	Byte 1):	1	1	1	1	1	1	0	1	FDH
	Byte 2):	1	0	0	1	0	1	1	0	96H
	Byte 3):	----------d-----------Offset								

r puede ser:

A - 111	E - 011
B - 000	H - 100
C - 001	L - 101
D - 010	

	Antes:	Después:
Código objeto	**Registros**	**Registros**
900CH FD	A FF / F 00	A 7F / F 2A
900DH 96	B / C	B / C
900EH 04	D / E	D / E
	H / L	H / L
	IX	IX
	IY 8976	IY 8976
	SP	SP
	PC 900C	PC 900F
	Memoria	**Memoria**
	897AH 80	897AH 80
	Flags activados:	Flags activados: N

XOR s — EXCLUSIVE OR entre acumulador y operando s.

Función: A <— A ⊕ s

Flags:

S	Z	-	H	-	P/V	N	C
X	X		0		X	0	0

Descripción: El contenido del acumulador A y el operando determinado a través de s, se interrelacionan bit a bit "EXCLUSIVE OR". El resultado es únicamente lógico 1 cuando exista desigualdad y se almacena en el acumulador.

Ejecución:

s	μs(4,00 Mhz)	Estados T	Ciclos M
r	1,00	4	1
n	1,75	7	2
(HL)	1,75	7	2
(IX + d)	4,75	19	5
(IY + d)	4,75	19	5

Direccionamiento: r: implícito/ n: inmediato/ (HL): indirecto/ (IX+d),(IY+d): indexado.

Formato: s: puede ser r,n,(HL),(IX+d) or (IY+d)

r:		1	0	1	0	1-----r---				
n:	Byte 1):	1	1	1	0	1	1	1	0	EEH
	Byte 2):	-----------n----------								Datos inmed.
(HL):		1	0	1	0	1	1	1	0	AEH
(IX + d):	Byte 1):	1	1	0	1	1	1	0	1	DDH
	Byte 2):	1	0	1	0	1	1	1	0	AEH
	Byte 3):	-----------d----------								Offset
(IY + d):	Byte 1):	1	1	1	1	1	1	0	1	FDH
	Byte 2):	1	0	1	0	1	1	1	0	AEH
	Byte 3):	-----------d----------								Offset

r puede ser:

A - 111	E - 011
B - 000	H - 100
C - 001	L - 101
D - 010	

Ejemplo: XOR (HL)

	Antes:	Después:
Código objeto	**Registros**	**Registros**
900FH: AE	A: 80, F: 00	A: 00, F: 44
	B: , C:	B: , C:
	D: , E:	D: , E:
	H: 89, L: 76	H: 89, L: 76
	IX:	IX:
	IY:	IY:
	SP:	SP:
	PC: 900F	PC: 9010
	Memoria	**Memoria**
	8976H: 80	8976H: 80
	Flags activados:	**Flags activados:** Z,P/V

COMMODORE



Ofrece un campo fascinante y amplio de problemáticas científicas. Para esto el libro contiene muchos listados interesantes: Análisis de Fornier y síntesis, análisis de redes, exactitud de cálculo, formateado de números, cálculo del valor PH, sistemas de ecuaciones diferenciales, modelo ladrón presa, cálculo de probabilidad, medición de tiempo, integración, etc.
64 en el campo de la Técnica y la Ciencia. 361 págs. P.V.P. 2.800,- ptas.



La obra Standard del floppy 1541, todo sobre la programación en disquettes desde los principiantes a los profesionales, además de las informaciones fundamentales para el DOS, los comandos de sistema y mensajes de error, hay varios capítulos para la administración práctica de ficheros con el FLOPPY, amplio y documentado Listado del Dos. Además un filón de los más diversos programas y rutinas auxiliares, que hacen del libro una lectura obligada para los usuarios del Floppy.
Todo sobre el Floppy 1541. Precio venta 3.200 ptas.



Un excelente libro, que le mostrará todas las posibilidades que le ofrece su grabadora de casettes. Describe detalladamente, y de forma comprensible, todo sobre el Datassette y la grabación en cassette. Con verdaderos programas fuera de serie: Autostart, Catálogo (¡busca y carga automáticamente!), backup de y a disco, SAVE de áreas de memoria, y lo más sorprendente: un nuevo sistema operativo de cassette con el 10-20 veces más rápido Fast Tape. Además otras indicaciones y programas de utilidad (ajuste de cabezales, altavoz de control).
El Manual del Cassette. 190 pág. P.V.P. 1.600,- ptas.



¡Por fin una introducción al código máquina fácilmente comprensible! Estructura y funcionamiento del procesador 6510, introducción y ejecución de programas en lenguaje máquina, manejo del ensamblador, y un simulador de paso a paso escrito en BASIC.
Lenguaje máquina para Commodore 64. 1984, 201 pág. P.V.P. 2.200,- ptas.



CONSEJOS Y TRUCOS, con más de 70.000 ejemplares vendidos en Alemania, es uno de los libros más vendidos de DATA BECKER. Es una colección muy interesante de ideas para la programación del Commodore 64, de POKEs y útiles rutinas e interesantes programas. Todos los programas en lenguaje máquina con programas cargadores en Basic.
64 Consejos y Trucos. 1984, 364 pág. P.V.P. 2.800,- ptas.



Este libro, contiene muchos interesantes programas de aprendizaje para solucionar problemas, descritos detalladamente y de manera fácilmente comprensible. Temas: progresiones geométricas, palanca mecánica, crecimiento exponencial, verbos irregulares, ecuaciones de segundo grado, movimientos de péndulo, formación de moléculas, aprendizaje de vocablos, cálculo de interés y su capitalización.
Manual escolar para su Commodore 64. 389 págs. P.V.P. 2.800,- ptas.



En el libro de los robots se muestran las asombrosas posibilidades que ofrece el CBM 64, para el control y la programación, presentadas con numerosas ilustraciones e intuitivos ejemplos. El punto principal: Cómo puede construirse uno mismo un robot sin grandes gastos. Además, un resumen del desarrollo histórico del robot y una amplia introducción a los fundamentos cibernéticos. Gobierno del motor, el modelo de simulación, interruptor de pantalla, el Port-Usuario cómodo del modelo de simulación, Sensor de infrarrojos, concepto básico de un robot, realimentación unidad cibernética, Brazo prensor, Oír y ver.
Robótica para su Commodore 64. 340 págs. P.V.P. 2.800 ptas.



Saberse apañar uno mismo, ahorra tiempo, molestias y dinero, precisamente problemas como el ajuste del floppy o reparaciones de la platina se pueden arreglar a menudo con medios sencillos. Instrucciones para eliminar la mayoría de perturbaciones, listas de piezas de recambio y una introducción a la mecánica y a la electrónica de la unidad de disco, hay también indicaciones exactas sobre herramientas y material de trabajo. Este libro hay que considerarlo en todos sus aspectos como efectivo y barato.
Mantenimiento y reparación del Floppy 1541. 325 págs. P.V.P. 2.800,- ptas.



Este es el libro que buscaba: un diccionario general de micros que contiene toda la terminología informática de la A a la Z y un diccionario técnico con traducciones de los términos ingleses de más importancia - los DICCIONARIOS DATA BECKER prácticamente son tres libros en uno. La increíble cantidad de información que contienen, no sólo los convierte en enciclopedias altamente competente, sino también en herramientas indispensables para el trabajo. El DICCIONARIO DATA BECKER se edita en versión especial para APPLE II, COMMODORE 64 e IBM PC.
El diccionario para su Commodore 64. 350 pág. P.V.P. 2.800,- ptas.



Casi todo lo que se puede hacer con el Commodore 64, está descrito detalladamente en este libro. Su lectura no es tan sólo tan apasionante como la de una novela, sino que contiene, además de listados de útiles programas, sobre todo muchas, muchas aplicaciones realizables en el C64. En parte hay listados de programas listos para ser tecleados, siempre que ha sido posible condensar «recetas» en una o dos páginas. Si hasta el momento no sabía que hacer con su Commodore 64, ¡después de leer este libro lo sabrá seguro!
El libro de ideas del Commodore 64. 1984, más de 200 páginas, P.V.P. 1.600,- ptas.



¿Ud. ha logrado iniciarse en código máquina? Entonces el «nuevo English» le enseñará cómo convertirse en un profesional. Naturalmente con muchos programas ejemplo, rutinas completas en código máquina e importantes consejos y trucos para la programación en lenguaje máquina y para el trabajo con el sistema operativo.
Lenguaje máquina para avanzados CBM 64. 1984, 206 pág. P.V.P. 2.200 ptas.



Este libro ofrece una amplia práctica introducción en el importante tema de la gestión de ficheros y bancos de datos, especialmente para los usuarios del Commodore 64. Con muchas interesantes rutinas y una confortable gestión de ficheros.
Todo sobre bases de datos y gestión de ficheros para Commodore-64. 221 págs. P.V.P. 2.200,- ptas.



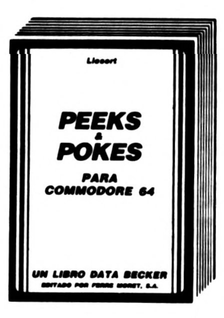
Gráficos para el Commodore 64 es un libro para todos los que quieren hacer algo creativo con su ordenador. El contenido abarca desde los fundamentos de la programación de gráficos hasta el diseño asistido por ordenador (CAD).
Gráficos para el Commodore 64. 295 pags. P.V.P. 2.200,- ptas.



Para los usuarios que posean un VIC-20, C-64 o PC-128 este libro contiene gran cantidad de consejos, trucos, listados de programas, así como información sobre Hardware, tanto si usted dispone de una impresora de margarita o de matriz, como si tiene un Plotter VC-1520, el GRAN LIBRO DE IMPRESORAS constituye una inestimable fuente de información.
Todo sobre Impresoras. 361 págs. P.V.P. 2.800,- ptas.



Con más de 60.000 ejemplares vendidos, ésta es la obra estándar para el COMMODORE 64. Todo sobre la tecnología, el sistema operativo y la programación avanzada del C-64. Con listado completo y exhaustivo de la ROM, circuitos originales documentados y muchos programas. ¡Conozca su C-64 a fondo!
64 Interno. 1984, 352 pág. P.V.P. 3.800,- ptas.



Con importantes comandos PEEK y POKE se pueden hacer también desde el Basic muchas cosas, para las que se necesitarían normalmente complejas rutinas en lenguaje máquina. Con una enorme cantidad de POKEs importantes y su posible aplicación. Para ello se explica perfectamente la estructura del Commodore 64: Sistema operativo, interpretador, página cero, apuntadores y stacks, generador de caracteres, registros de sprites, programación de interfaces, desactivación de interrupt. Además una introducción al lenguaje máquina. Muchos programas ejemplo.
PEEKS y POKES. 177 pág. P.V.P. 1.600,- ptas.



Este libro presenta una detallada e interesante introducción a la teoria, conceptos básicos y posibilidades de uso de la inteligencia artificial (1A). Desde un resumen histórico sobre las máquinas «pensantes» y «vivientes» hasta programas de aplicación para el Commodore 64.
Inteligencia artificial. 395 págs. 2.800,- ptas.



64, Consejos y Trucos vol. 2 contiene una gran profusión de programas, estímulos y muchas rutinas útiles. Un libro que constituye una ayuda imprescindible para todo aquél que quiera escribir programas propios con el COMMODORE.
Consejos y Trucos, Commodore 64. Vol. 2. 259 págs. 2.200,- ptas.



Este libro ofrece al programador interesado una introducción fácilmente comprensible para los tan extendidos Assembler PROFI-ASS, SM MAE y T.E-X.ASS. con consejos y trucos de gran utilidad, indicaciones y programas adicionales. Al mismo tiempo sirve de manual orientado a la práctica, con aclaraciones de conceptos importantes e instrucciones.
El Ensamblador. 250 páginas. 2.200,- ptas.

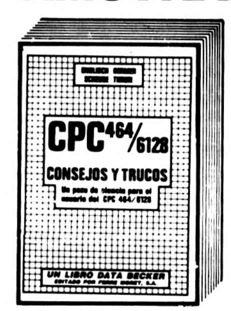


El libro de Primicias del Commodore 128 no ofrece solamente un resumen completo de todas las características y rendimientos del sucesor del C-64 y con ello una importante ayuda para su adquisición. Muestra, además, todas las posibilidades del nuevo equipo en función de sus tres modos de operación.
Todo sobre el nuevo Commodore 128. 250 págs. P.V.P. 2.200,- ptas.



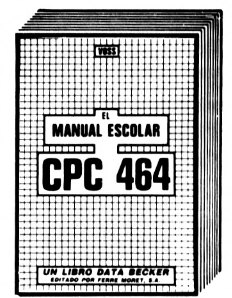
El libro Commodore 128-Consejos y Trucos es un filón para cualquier poseedor del C-128 que desee sacar más partido a su ordenador. Este libro no sólo contiene gran cantidad de programas-ejemplo, sino que además explica de un modo sencillo y fácil la configuración del ordenador y de su programación.
Commodore 128-Consejos y Trucos. 327 págs. 2.800,- ptas.

AMSTRAD



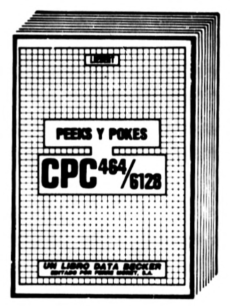
Ofrece una colección muy interesante de sugerencias, ideas y soluciones para la programación y utilización de su CPC-464: Desde la estructura del hardware, sistema de funcionamiento - Tokens Basic, dibujos con el joystick, aplicaciones de ventanas en pantalla y otros muchos interesantes programas como el procesamiento de datos, editor de sonidos, generador de caracteres, monitor de código máquina hasta listados de interesantes juegos.

CPC-464 Consejos y Trucos. 263 págs. P.V.P. 2.200,- ptas.



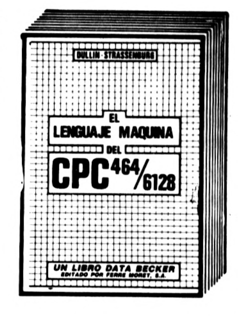
Escrito para alumnos de los últimos cursos de EGB y de BUP, este libro contiene muchos programas para resolver problemas y de aprendizaje, descritos de una forma muy compleja y fácil de comprender. Teorema de Pitágoras, progresiones geométricas, escritura cifrada, crecimiento exponencial, verbos irregulares, igualdades cuadráticas, movimiento pendular, estructura de moléculas, cálculo de interés y muchas cosas más.

CPC-464 El libro del colegio. 380 págs. P.V.P. 2.200,- ptas.



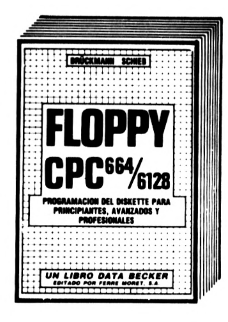
PEEKS, POKES y CALLS se utilizan para introducir al lector de una forma fácilmente accesible al sistema operativo y al lenguaje máquina del CPC. Proporciona además muchas e interesantes posibilidades de aplicación y programación de su CPC.

PEEKS y POKES del CPC 464/6128. 180 pág. P.V.P. 1.600,- ptas.



El libro del lenguaje máquina para el CPC 464/6128 está pensado para todos aquellos a quienes no les resulta suficiente con las posibilidades y rapidez del BASIC. Se explican aquí detalladamente las bases de la programación en lenguaje máquina, el funcionamiento del procesador Z-80 con sus respectivos comandos así como la utilización de las rutinas del sistema con abundantes ejemplos. El libro contiene programas completos de aplicación tales como Ensamblador. Desensamblador y Monitor, facilitando de esta manera la introducción del lector en el lenguaje máquina.

El Lenguaje Máquina del CPC 464/6128. 330 pág. P.V.P. 2.200,- ptas.



El LIBRO DEL FLOPPY del CPC lo explica todo sobre la programación con discos y la gestión relativa de ficheros mediante el floppy DDI-1 y la unidad de discos incorporada del CPC 664/6128. La presente obra, un auténtico stándart, representa una ayuda incomparable tanto para el que desee iniciarse en la programación con discos cómo para el más curtido programador de ensamblados. Especialmente interesante resulta el listado exhaustivamente comentado del DOS y los muchos programas de ejemplo, entre los que se incluye un completo paquete de gestión de ficheros.

El Libro del Floppy del CPC. 353 pág. P.V.P. 2.800,- ptas.



¡Dominar CP/M por fin! Desde explicaciones básicas para almacenar números, la protección contra la escritura, o ASCII, hasta la aplicación de programas auxiliares de CP/M, así como «CP/M interno» para avanzados, cada usuario del CPC rápidamente encontrará las ayudas e informaciones necesarias, para el trabajo con CP/M. Este libro tiene en cuenta las versiones CP/M 2.2, así como CP/M Plus (3.0), para el AMSTRAD CPC 464, CPC 664 y CPC 6128.

CP/M. El libro de ejercicios para CPC. 260 pág. P.V.P. 2.800,- ptas.





Escrito para alumnos de los últimos cursos de EGB y de BUP, este libro contiene muchos programas para resolver problemas y de aprendizaje, descritos de una forma muy completa y fácil de comprender. Teorema de Pitágoras, progresiones geométricas, escritura cifrada, crecimiento exponencial, verbos irregulares, igualdades cuadráticas, movimiento pendular, estructura de moléculas, cálculo de interés y muchas cosas más.
MSX el Manual Escolar. 389 págs. P.V.P. 2.800,- ptas.



El libro contiene una amplia colección de importantes programas que abarcan, desde un desensamblador hasta un programa de clasificaciones deportivas. Juegos superemocionantes y aplicaciones completas. Los programas muestran además importantes consejos y trucos para la programación. Estos programas funcionan en todos los ordenadores MSX, así como en el SPECTROVIDEO 318 328.
MSX Programas y Utilidades, 1985, 194 pág. P.V.P. 2.200,- ptas.



Las computadoras MSX no sólo ofrecen una relación precio/rendimiento sobresaliente, sino que también poseen unas cualidades gráficas y de sonido excepcionales. Este libro expone las posibilidades de los MSX de forma completa y fácil. El texto se completa con numerosos y útiles programas ejemplo.
MSX Gráficos y Sonidos, 250 págs. P.V.P. 2.800,- ptas.



Este libro contine una colección sin igual de trucos y consejos para todos los ordenadores con la nueva norma MSX. No sólo contiene las recetas completas, sino también los conocimientos básicos necesarios.
MSX - Consejos y Trucos. 288 págs. P.V.P. 2.200,- ptas.



El libro del Lenguaje Máquina para el MSX está creado para todos aquellos a quienes el BASIC se les ha quedado pequeño en cuanto a rendimiento y velocidad. Desde las bases para la programación en Lenguaje Máquina, pasando por el método de trabajo del Procesador Z-80 y una exacta descripción de sus órdenes, hasta la utilización de rutinas del sistema todo ello ha sido explicado en detalle e ilustrado con múltiples ejemplos en este libro.
El libro contiene, además, como programas de aplicación, un ensamblador un desensamblador y un monitor.
MSX Lenguaje Máquina. 306 págs. 2.200,- ptas.

ZX SPECTRUM

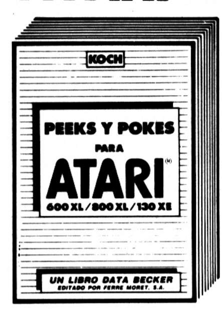


Una interesante colección de sugestivas ideas y soluciones para la programación y utilización de su ZX ESPECTRUM. Aparte de muchos peeks, pokes y USRs hay también capítulos completos para, entre otros, entrada de datos asegurado sin bloqueo de ordenador, posibilidades de conexión y utilización de microdrives y lápices ópticos, programas para la representación de diagramas de barra y de tarta, el modo de utilizar óptimamente ROM y RAM.
ZX Spectrum Consejos y Trucos, 211 pág. P.V.P. 2.200,- ptas.



Escrito para alumnos de los últimos cursos de EGB y de BUP, este libro contiene muchos programas para resolver problemas y de aprendizaje, descritos de una forma muy completa y fácil de comprender. Teorema de Pitágoras, progresiones geométricas, escritura cifrada, crecimiento exponencial, verbos irregulares, igualdades cuadráticas, movimiento pendular, estructura de moléculas, cálculo de interés y muchas cosas más.
ZX Spectrum el Manual Escolar. 389 págs. P.V.P. 2.200,- ptas.

ATARI



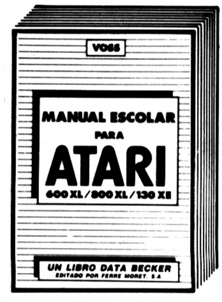
Tan interesante como el tema, es el libro que explica de forma fácilmente comprensible el manejo de Peeks y Pokes importantes, y representa un gran número de Pokes con sus posibilidades de aplicación, incluyendo además programas ejemplo. Al lado de temas como lo son la memoria de la pantalla, los bits y los bytes, el mapa de la memoria, la tabla de modos gráficos o el sonido, también se detalla de forma magnífica la estructura del ATARI 600XL/800XL/130XE.
Peeks y Pokes para ATARI 600XL/800XL/130XE. 251 pág. P.V.P. 2.200, ptas.



Una lograda introducción al sugestivo tema de los «juegos estratégicos» Desde juegos sencillos con estrategia fija a juegos complejos con procedimientos de búsqueda hasta programas con capacidad de aprendizaje —muchos ejemplos interesantes, escritos por supuesto de forma fácilmente comprensible. Con programas de juegos ampliamente detallados: NIM con un montón, bloqueo, hexapawn, mini-damas y muchos más.
Juegos estratégicos y cómo programarlos en el ATARI 600XL/800XL/130XE. 181 pág. P.V.P. 1.600,- ptas.



Jugar a aventuras con éxito y programarlas uno mismo - todo lo verdaderamente importante sobre el tema, lo contiene este guía fascinante que te lleva a través del mundo fantástico de las aventuras. El libro abarca todo el espectro, hasta las más sofisticadas aventuras gráficas llenas de trucos, acompañándolas siempre de numerosos programas ejemplo. Sin embargo la clave —al margen de muchas aventuras para teclear— es un generador de aventuras completo, mediante el cual la programación de aventuras se convierte en un juego de niños.
Aventuras - y cómo programarlas en el ATARI 600XL/800XL/130XE. 284 pág. P.V.P. 2.200,- ptas.



Muchos programas interesantes de soluciones de problemas y de aprendizaje, descritos de forma amplia y comprensible, y adecuados sobre todo para escolares. ¡Aquí el aprendizaje intensivo se convierte de una tarea divertida! Al margen de temas como los verbos irregulares, o las ecuaciones de segundo grado, un resumen corto de las bases del tratamiento electrónico de datos, y una introducción a los principios del análisis de problemas, completan este libro que debería obrar en posesión de cualquier escolar.
El libro escolar para ATARI 600XL/800XL/130XE. 389 pág. P.V.P. 2.800,- ptas.

OTROS TITULOS



El primer libro recomendado para escuelas de enseñanza de informática y para aquellas personas que quieren aprender la programación. Cubre las especificaciones del Ministerio de Educación y Ciencia para Estudios de Informática. Es el primer libro que introduce a la lógica del ordenador. Es un elemento de base que sirve como introducción para la programación en cualquier otro lenguaje. No se requieren conocimientos de programación ni siquiera de informàtica. Abarca desde los métodos de programación clásicos a los más modernos.
Metodología de la Programación. 250 págs. P.V.P. 2.200,- ptas.



La técnica y programación del Procesador Z80 son los temas de este libro. Es un libro de estudio y de consulta imprescindible para todos aquellos que poseen un Commodore 128, CPC, MSX u otros ordenadores que trabajan con el Procesador Z80 y desean programar en lenguaje máquina.
El Procesador Z80. 560 pág. P.V.P. 3.800,- ptas.



El tema de este libro es la técnica y programación de los procesadores de la familia 68000. Es una obra de consulta indispensable, un manual para todo programador que quiera utilizar las ventajas del 68000.
Técnica y programación para el procesador 68000. 516 págs. P.V.P. 3.800,- ptas.

EL CONTENIDO:

La técnica y programación del Procesador Z80 son los temas de este libro. Es un libro de estudio y de consulta imprescindible para todos aquellos que poseen un Commodore 128, CPC, MSX u otros ordenadores que trabajan con el Procesador Z80 y desean programar en lenguaje máquina.

Extracto del contenido:

- Arquitectura del sistema
- Descripción de los PIN
- Registros
- Ejecución de instrucciones
- Flags
- CPU-Software
- Conexión de elementos del sistema
- Transmisión de datos en serie/paralelos
- Elementos del contador y del Timer del Z80-CPC con instrucciones

ESTE LIBRO HA SIDO ESCRITO POR:

Otto Hausbacher ingeniero y especialista del Z80. Además de la actividad como programador de máquinas, desarrolla sistemas de mando mediante procesadores Z80.

ISBN 84-86437-32-6

Hausbacher / El Procesador Z80


AMSTRAD
CPC
OCR
MÉMOIRE ÉCRITE
MEMORY ENGRAVED
MEMORIA ESCRITA
https://acpc.me/